TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

BAKALÁŘSKÁ PRÁCE

Apoštol Ondřej

Liberec 2007

TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

Studijní program: **B2612 – Elektrotechnika a informatika**

Studijní obor: **2612R011 – Elektronické informační a řídící systémy**

Řízení malého stejnosměrného motorku

BAKALÁŘSKÁ PRÁCE

Drive of a small direct current motor

BACHELOR THESIS

Autor: Ondřej Apoštol

Vedoucí bakalářské práce: ing. Tomáš Martinec

Konzultant: ing. Pavel Pirkl

Liberci dne 18.5.2007

Prohlášení

Byl jsem seznámen s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé BP a prohlašuji, že s o u h l a s í m s případným užitím mé diplomové práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Bakalářskou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

V Liberci dne ....

podpis

Poděkování

Rád bych poděkoval všem, kteří mi pomohli při vzniku této práce. Především svému vedoucímu bakalářské práce Ing. Tomáši Martincovi za konzultace a odbornou pomoc. Dále panu Ing. Leoši Petržílkovi za vyhotovení desek plošných spojů.

Abstrakt

Tématem bakalářské práce byl návrh a realizace zpětnovazebního řízení malého stejnosměrného motorku. Práce se zabývá nejen hardwarovým návrhem a řešením, ale i obslužným softwarem, který umožňuje řídit motorek a snímat jeho chování pomocí PC. Práce je tématicky dělena do tří částí. V první části převažuje obecný popis motorů a způsoby jejich řízení. V druhé části je podrobně popsán hardware včetně vlastností součástek a návrhů desek plošných spojů. Poslední část popisuje realizaci softwaru.

Abstract

The theme of my Bachelor theses was design and implementation of a small feedback engine. My thesis considers not only hardware design and its analysis, but also utility software which allows engine controlling and scanning its behaviour by PC. The thesis is divided into three parts. First one deals with general description of motors and of ways they can be controlled. In the second part, hardware is characterized in detail, including features of all components and design of cards with printed circuits. The last part describes implementation of software.

Obsah

Klíčová slova

uP – mikroprocesor

ss motorek – stejnosměrný motorek

PŘJ – programovatelná řídící jednotka

ATMEL – mikroprocesor konkrétního výrobce

PWM – pulsně šířková modulace

RS232 – sériová linka

ASCII tabulka – tabulka používaných znaků ve výpočetní technice

PC – personal computer (osobní počítač)

DPS – deska plošného spoje

Úvod

Cílem bakalářské práce bylo navrhnout zpětnovazební řízení malého stejnosměrného motorku pomocí mikroprocesoru. Konkrétně se jedná o řízení dvou stejnosměrných motorků, snímaní jejich polohy a rychlosti otáčení.

Dalším cílem bylo realizovat potřebné programové vybavení pro ovládací panel v PC. Ten se skládá z komunikačního protokolu realizující kódování a dekódování dat pro komunikaci po sériové lince a ze samotného ovládacího panelu sloužícího k řízení chodu a rychlosti obou motorků a k odečítání jejich rychlostí a směrů.

Poslední úkol bylo navrhnout potřebné programové vybavení pro řídící mikroprocesor, které umožní všechny řídící funkce a požadované přenosy.

1. Motory

1.1 Elektromotor

Elektromotor je stroj, který nám umožňuje měnit elektrickou energii na mechanickou práci. Opačnou přeměnu, tedy změnu mechanické energie na elektrickou provádí zařízení zvané generátor popř. dynamo. Častou jsou tyto zařízení téměř identická (malé konstrukční detaily).

Většina elektromotorů pracuje na elektromagnetickém principu, ale existují i motory založené na jiných elektromechanických jevech jako je elektrostatická síla, pizoelektrický efekt či tepelné účinky proudu. Základním principem, na němž jsou elektromagnetické motory založeny, je vzájemné působení vodičů, kterými protéká proud. Tuto sílu také popisuje Lorentzův zákon síly.

V běžném rotačním motoru je umístěn rotor tak, aby vodiče rotoru a magnetické pole statoru vyvíjely točivý moment kolem osy rotoru. Působením tohoto momentu tak vzniká síla, která způsobuje otáčení rotoru.

Většina elektrických motorů je rotačních, ale existují i jiné motory, jako je lineární elektromotor. V rotačním motoru se rotující část (obvykle se nachází uvnitř) nazývá rotor a statická část (část, která je pevně přichycena) se nazývá stator. Motor obsahuje pevně spojenou sadu elektromagnetů umístěných obvykle na rotoru. I když se tento rám obvykle nazývá kotva, jde o nesprávné užití termínu. Kotva je ta část motoru, která koná práci, nebo ta část generátoru, přes kterou se generuje výstupní napětí. Podle typu motoru může jako kotva sloužit jak stator tak i rotor.

1.2 Motory na stejnosměrný proud (SS motory)

Historie: Jeden z prvních rotačních elektromotorů (zřejmě první) vynalezl Michael Faraday z roce 1821. Motor se skládal z volně zavěšeného drátu ponořeného do nádrže s rtutí. Ve středu nádrže byl umístěn permanentní magnet. Elektrický proud procházel drátem, drát rotující kolem magnetu pak prokazoval, že proud vytvořil otáčivé magnetické pole kolem drátu. První moderní motor na stejnosměrný proud byl náhodně objeven v roce 1873, když jistý Zénobe Gramme vodivě spojil roztočené dynamo s druhým stojícím dynamem, z něhož se stal napájený motor.

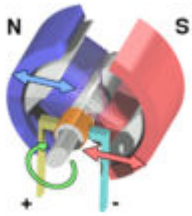
1.3 Motor s permanentním magnetem

Jedná se o nejjednodušší motor na stejnosměrný proud. Stator má tvořen permanentím magnetem a rotující kotvu ve formě elektromagnetu se dvěma póly. Rotační přepínač zvaný komutátor mění směr elektrického proudu procházejícího kotvou dvakrát během jedné otáčky. Tím se zajistí, že síla působící na póly rotoru má stále stejný směr. V okamžiku přepnutí polarity udržuje běh tohoto motoru ve správném směru setrvačnosti.

Motory s permanentním magnetem se dodnes používají např. v modelářství. Jen kotva ke minimálně třípólová, aby nevznikal problém s tzv. mrtvým úhlem motoru. Výhodou motoru s permanentním magnetem je možnost snadné změny směru otáčení pomocí polarity vstupního napětí (pokud na vývody motoru přivedeme napětí roztočí se na jednu stranu v okamžiku prohození přiváděcích vodičů se motor roztočí na stranu druhou).

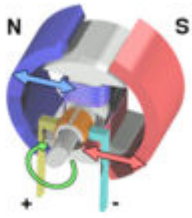
Komutátor zajišťuje, že se v cívce změní směr proudu + a - (resp. - a +) po každém pootočení o 180° (platí pouze pro dvoupólový motor). Takto dochází ke změně indukčních siločar v cívce.

1.4 Funkce stejnosměrného motoru



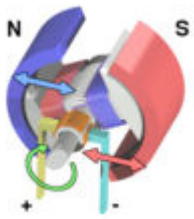
Rotor (kotva) je přes oranžový komutátor připojen ke zdroji stejnosměrného napětí. Stator je tvořen dvěma permanentními magnety.

Obr.1 – První fáze



Vzhledem k polaritě statoru a rotoru se souhlasné póly (barvy) odpuzují a tím vzniká otáčivá síla (rotor se otáčí).

Obr.2 – Druhá fáze



Opačné póly se přitahují, rotor se stále otáčí. V okamžiku, kdy se rotor dostane do vodorovné polohy, dojde na akumulátoru k přepnutí polarity magnetického pole rotoru.

Obr.3 – Třetí fáze
Použitá literatura [5]

1.5 Obecné vlastnosti stejnosměrných motorů

Rychlost motoru na stejnosměrný proud obecně závisí na velikosti napětí a proudu procházejících vinutím motoru a na zátěži neboli velikosti brzdného momentu. Rychlost motoru při daném brzdném momentu je úměrná napětí a točivý moment je úměrný proudu. Rychlost motoru lze regulovat změnou vstupního napětí.

Výhodou stejnosměrných motorů je relativní jednoduchost a univerzálnost využití. Sériový a derivační motor mohou fungovat nejen na stejnosměrný, ale i střídavý proud nízkých frekvencí. Jsou to tedy motory univerzální. Další výhodou proti motorům střídavým je možnost dosáhnout libovolných mechanicky dosažitelných otáček (motory na střídavý proud mají obvykle otáčky omezeny frekvencí sítě - 50Hz = 3000 ot/min). Proto tyto motory nacházejí uplatnění v takových strojích, jako jsou vrtačky, mixéry, ale třeba i automobily a lokomotivy.

Největší nevýhodou stejnosměrných motorů je existence komutátoru. Je to mechanický přepínač, který spíná velké proudy a je kromě náchylnosti k poruchám také náročný na údržbu a seřízení. Jiskření na kartáčcích (obvykle jsou tvořeny bloky uhlíku) je zdrojem významného elektrického rušení. S rozvojem levnější a spolehlivější elektroniky jsou proto stejnosměrné motory postupně vytlačovány motory s rotujícím magnetickým polem buzeným elektronicky.

1.6 Reverzace chodu stejnosměrných motorů

U stejnosměrných motorů plně postačuje přepólovaní a můžeme s ním bezproblémově měnit chod otáčení motorku.

1.7 Další druhy motorů

Motory na střídavý proud : - Synchronní motor
- Asynchronní motor

Krokový motor

Lineární elektromotor

Střídavý servomotor

1.8 Řízení stejnosměrných motorů

Řízení je cílevědomá činnost, při níž se hodnotí a zpracovávají informace o řízeném objektu nebo procesu i informace o dějích vně tohoto procesu, a podle nich se ovládají příslušná zařízení tak, aby bylo dosaženo předepsaného cíle.

Jak již vyplývá ze zadání, motorek bylo nutné řídit zpětnovazebně. Řízení samotného motorku je možné několika způsoby. Velmi rozšířený způsob řízení je pomocí elektronických součástek, např. pomocí potenciometru, jenž nám umožňuje měnit odpor, resp. velikost napětí, které vstupuje do motoru. U tohoto způsobu řízení je však poměrně složité měnit směr otáčení motoru, které většinou musíme realizovat pomocí přepínačů. Další nevýhodou je, že při častějších zásazích do regulace motorku, dochází k poměrně velkému opotřebení součástek, které je tedy nutno často kontrolovat a také měnit. Proto je výhodnější řídit motorek pomocí spínání a rozpínání polovodičových součástek. V praxi můžou být použity součástky jako jsou diody, triaky či tranzistory. Asi nejvyužívanější součástkou jsou tranzistory. Při řízení motorku používáme právě čtyři tranzistory a jejich typické zapojení se nazývá H můstek, který je podrobně popsán v kapitole 2.4. Samotný H můstek je však nutné také ovládat. Pokud by jsme na jednotlivé tranzistory H můstku pouze přiváděli napětí, mohli bychom pouze měnit směr otáčení a rychlost otáček by byla buď 0 nebo MAX (samozřejmě by napětí muselo být přivedeno na správné tranzistory). Proto se při řízení H můstku používá jev nazývaný jako pulsně šířková modulace (kapitola 2.4.2), pomocí ní lze měnit jak směry, tak i samotnou rychlost otáčení motorku. Právě tímto způsobem jsem se rozhodl řídit motorek v bakalářské práci. Motorek je připojen na H můstek, který je pomocí mikroprocesoru řízen pulsně šířkovou modulací .

Zpětnovazební řízení znamená, že signály a řídící data nejen dodáváme, ale zároveň dostáváme nazpět. U řízení motorku je asi nejlogičtější zabývat se daty okolo chodu motorku, tzn. zda-li běží, jakou rychlostí a na jakou stranu. Tyto data můžeme získávat mnoha metodami a přístroji.

1.8.1. Metody a přístroje pro snímání polohy

Metody

- Kontaktní
- Bezkontaktní

- *Kontaktní metody:* Jak již vyplývá z názvu tato metoda funguje na principu přímého kontaktu mezi měřícím přístrojem a pohyblivou částí motoru (rotoru).
- *Bezkontaktní metody:* Principielně nemají žádný přímý fyzický kontakt mezi měřícím přístrojem a rotorem (snímaní pomocí laseru, infasvětla atd.).

Přístroje

- Mechanické
- Elektromagnetické (tachodynamo)
- Optoelektronické (fotodioda, fototranzistor, fotoodpor)
- Magnetické (Hallova sonda)
- Jiné: změna kapacity, indukčnosti, stroboskopicky

Elektromagnetické otáčkoměry: Permanentní magnet se otáčí uvnitř hliníkového hrníčku, kde vířivé proudy vytváří točivý moment, a hrníček, jež je spojen s ukazující ručičkou a pružinou, s sebou strhávají a ručička ukazuje výchylku jež je lineárně úměrná rychlosti otáčení. Často se užívají v automobilech.

Tachodynamo – Stejnosměrný stroj uzpůsobený pro měření otáček. Permanentní magnet, speciální kolektor a kartáče, homogenní magnetické pole, malé zvlnění, kotva železa, velký výstupní odpor.

Tachoalternátor: Střídavý stroj. Nemá komutátor, rotor je z permanentního magnetu (pólové dvojice). Velká vzduchová mezera (aby se nezeslabily magnety při zkratu) velká robustnost, měří špatně malé rychlosti, mění se U i f.

Snímač s fotoelektrickým čidlem:

- Průchozí metoda – Zdroj světla (žárovka) vysílá záření, které prochází skrz děrovaný rotační kotouč. Za ním se nachází senzor, který zareaguje vždy, když projde záření.
- Odrazová metoda – Zdroj světla (infradioda) vysílá záření, které se od nějaké odrazivé plochy odrazí zpět a ovlivní tím snímač (fotoodpor aj.). Frekvence impulsů ze snímače je úměrná měřené rychlosti u obou způsobů.

Magnetické: Kotouč má zuby – změna magnetického toku v cívce je zpracována do el. impulsů a ty čítány. Požívají se často v automobilech snímání otáčení kol pro ABS systém.

Změna kapacity: Např.Snímač s Hallovou sondou. Nahradí-li se zuby magnety, lze tyto magnety (zuby) snímat Hallovou sondou s dvoustavovým výstupem – pulzující napětí. Často u digitálních tachometrů na jízdních kolech.

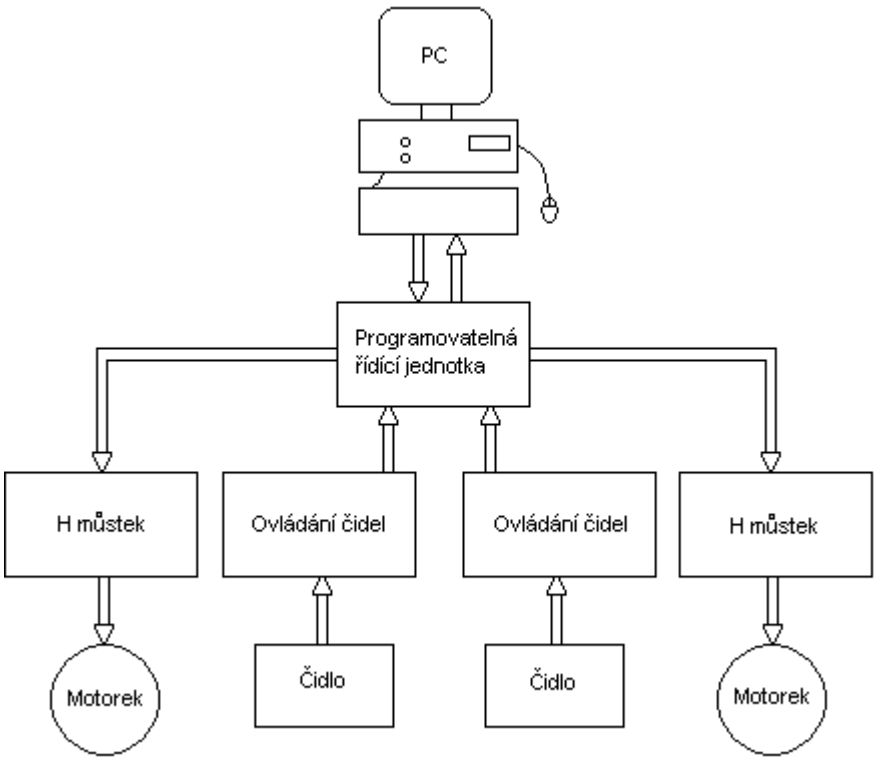
Stroboskopicky: Časový výběr polohy. Blikajícím světlem (zábleskem) osvětlujeme měřené místo (rotační těleso) a při rovnosti násobku blikání (záblesků) se nám obraz rotujícího kotouče zastaví a tím můžeme určit otáčky ,případné zpoždění. Nutno měřit od větší frekvence – od shora.

Jako nejlepší varianta se jevila bezkontaktní metoda. Kontaktní metoda se používá spíše u velkých motorů. S bezkontaktních metod je výběr mezi stroboskopickou metodou a pomocí optoelektronických součástek. U stroboskopické metody by bylo pro řídící logiku poměrně těžké určit, kdy se odraz rotujícího kotouče jeví jako zastavený. Další problém je s určováním násobků frekvencí. Proto jsem se rozhodl použít optoelektronické součástky. Dále se jako výhodnější jevila možnost odrazivé metody, protože zabere méně místa nežli neodrazivá. Stačí totiž pouze jeden plošný spoj obsazený jak vysílačem, tak i přijímačem. U neodrazivé metody by bylo nutno použít dva plošné spoje, jeden jako vysílač a druhý jako přijímač. Tato metoda by také zabrala o dvě vstupně / výstupní svorkovnice navíc.

2. Hardware

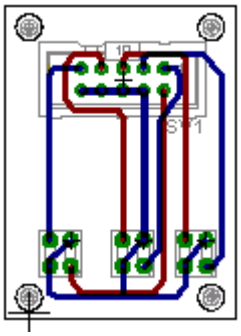
Následující část se podrobně zabývá návrhem hardwaru a popisem samotné realizace. Hardware se skládá z modulů: Čidla, Ovládání čidel, Programovatelné řídící jednotky (dále jen PŘJ) a H můstku. Mimo PŘJ je zbytek modulů obsažen dvakrát, aby bylo možné řídit dva motorky současně. Všechny moduly jsou podrobněji rozepsány v této kapitole, včetně obrázků návrhů desek plošných spojů. Schémata jsou přiložena v příloze.Většinu návrhů desek plošných spojů jsem nenavrhoval (mimo h můstku) a byly již vytvořeny mým předchůdcem.

Návrhy desek plošných spojů a schémata byly vytvořeny pomocí programu EAGLE 4.16r2. Jedná se o volně šiřitelný software, který je ve free verzi omezen pouze velikostí Boardu (velikost desky plošného spoje), jinak je plně funkční. Blokové schéma hardwarového návrhu a vzájemné propojení mezi jednotlivými moduly je vidět na *Obr.4* . Šipky představují směr toku dat, resp. signálů. Schéma také zobrazuje, jak jsou moduly vzájemně nadřazeny (nejvyšší PŘJ, nejnižší Čidla a motorky).



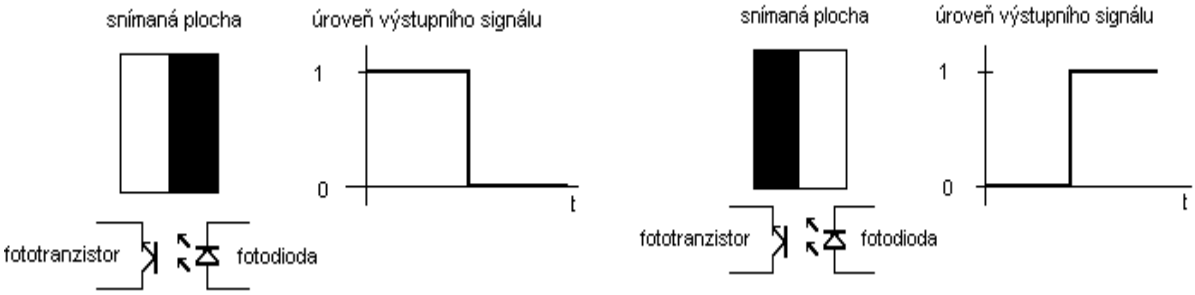
Obr.4 – Blokové schéma propojení a toku dat v hardwarovém návrhu

2.1 IR Čidla



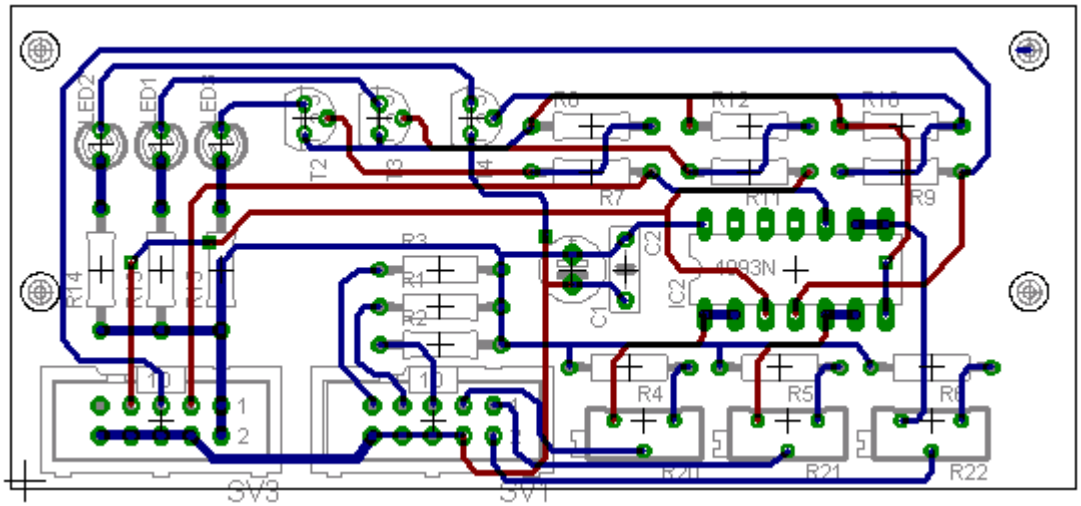
Obr.5 – Návrh desky plošného spoje

Tento plošný spoj je osazen odrazovými infrasenzory CNY70 . Jedná se o kombinaci infračervené LED diody a NPN fototranzistoru usazených vedle sebe v jednom pouzdře. Fototranzistor je součástka, která je schopna při dopadajícím záření do kolektorového PN přechodu vytvořit napětí mezi bází a emitorem, otevřít se a umožnit tak průchod proudu z připojeného zdroje. Fototranzistor reaguje na záření vysílané fotodiodou, které se odrazí od procházejícího bílého proužku. Černá barva toto záření pohltí a nedojde k otevření PN přechodu. Vzdálenost čidla od snímaného objektu musí být < 6 mm, vhodné je však použít vzdálenosti do 1,5 mm. Materiál pro odrazivou plošku musí být neprůhledný, protože sluneční záření by mohlo ovlivňovat fototranzistor. Pro snímaní rychlostí a směrů otáčení je nutno použít tři čidla. Dvě pro určení směru otáčení, jedno pro rychlost. Výstup ze svorkovnice SV1 vede na Ovládání čidel (vizte. kapitola 2.2).



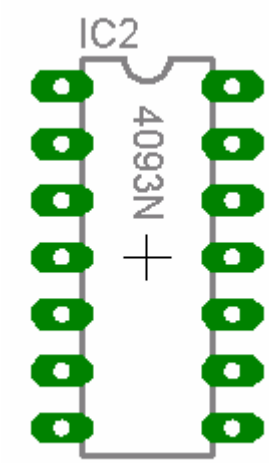
Obr.6 Snímaná plocha a úrovně signálů

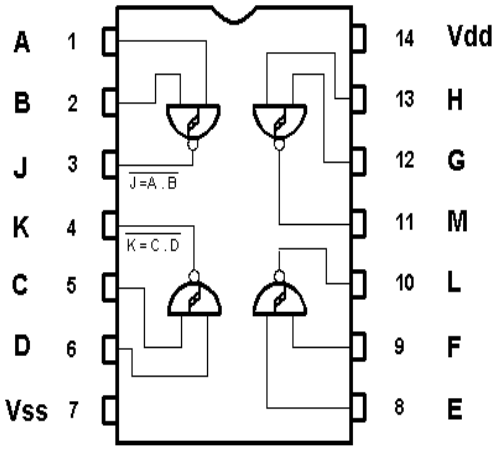
2.2 Vyhodnocení signálů z IR čidel



Obr.7 – Návrh desky plošného spoje

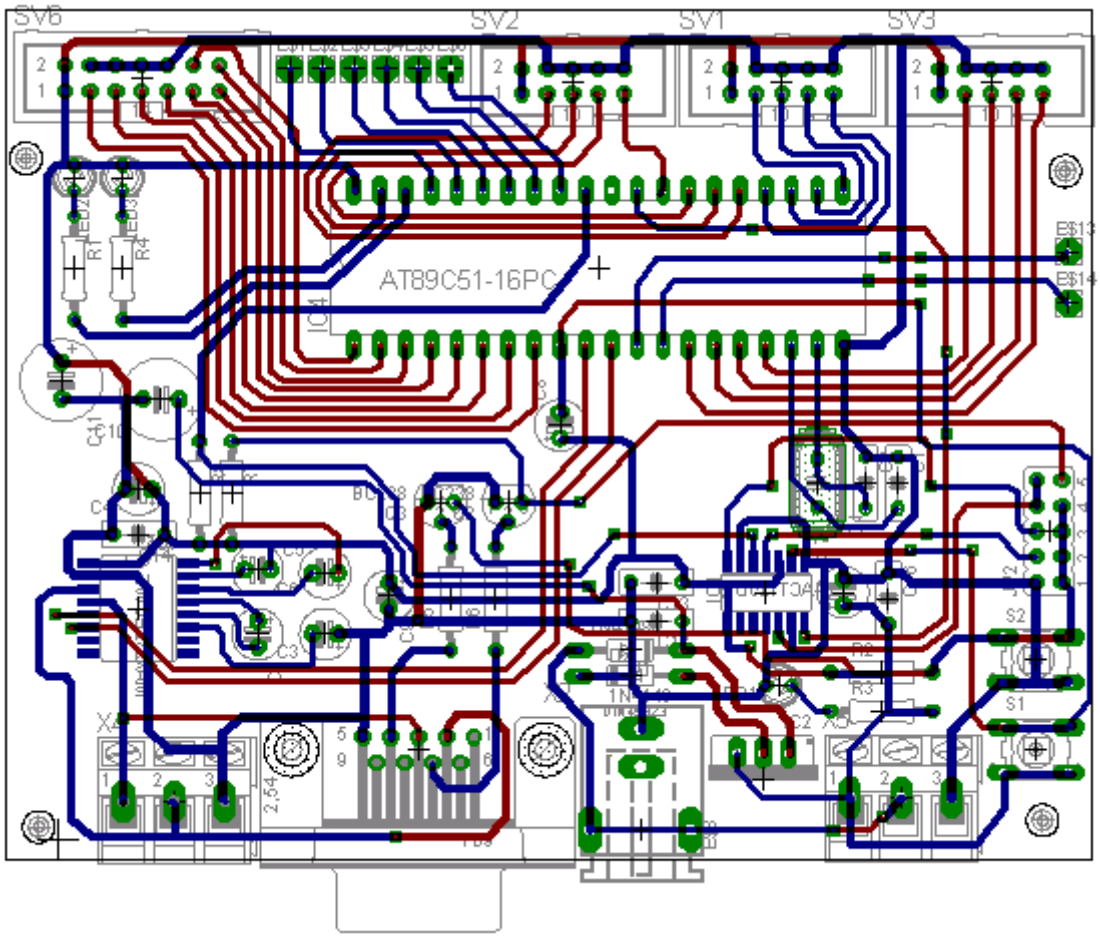
Tento plošný spoj slouží jako filtr pro signál, který vystupuje z čidel. Je osazen integrovaných obvodem HCF 4093BE, což je polovodičová součástka složená ze čtyř „schmitových“ spouštěcích obvodů. Jedná se o hradla typu NAND tzn., že provádí negaci signálu na vstupu hradel. Hradla mají vždy dva vstupy na jeden výstup. U našeho zapojení jsou využity pouze tři hradla NAND, čtvrté hradlo zůstalo neobsazeno. Každé hradlo je přivedeno na jiný snímač polohy. Odpory R1, R2 a R3 slouží pro regulaci proudu, který vstupuje do LED diod. Tyto LED diody slouží k indikaci signálu vystupujícího z čidel. Takže pokud infrasenzor zaznamená na indikačním kolečku (*Obr.6*) přechod barev bílá – černá naindukuje proud. Díky tomuto signálu se pak rozsvítí LED diody. Jezdce na trimrech R20, R21 a R22 vedou na kolektory tranzistorů v čidlech, které je v sepnutém stavu uzemňují. Např. při průchodu bílého proužku se tranzistor čidla sepne, uzemní výstupy hradla a na výstupu hradla NAND se objeví log.1. Při průchodu černého proužku naopak tranzistor proud nevede, vstup na hradlo NAND jde přes odpor a trimr je připojen na napájecí napětí Vcc. Tudíž se na výstupu hradla NAND objeví logická 0. U hradel NAND GATE přepíná v různých bodech pro kladné a záporné signály. Rozdíl mezi kladnou a zápornou voltáží je definován jako hysterézní voltáž. Výstup ze svorkovnice SV1 vede na čidlo a výstup ze svorkovnice SV3 vede na Programovatelnou řídící jednotku (vizte. kapitola 2.3).





Obr.8 – Pouzdro 4093 v provedení PID *Obr.9 – Zapojení PINů*

2.3 Programovatelná řídící jednotka (PŘJ)



Obr.10 – Návrh desky plošného spoje

Tento modul lze označit jako centrální jednotku. Slouží pro naprogramování mikroprocesoru ATMEL, dále ke komunikaci po sériové lince (vizte. kapitola 2.5) s PC, k propojení s ostatními moduly (vizte. kapitola 2.6) a je zde také přivedeno napájení 12V, které se indikuje pomocí led diody zelené barvy. Jedny z nejdůležitějších součástí jsou mikročip ATMEL, který se osazuje do 40 pinové patice a převodník MAX232. Mikročip ATMEL lze také dle potřeby nahradit jiným mikročipem, např. od firmy DALLAS.

U tohoto modulu se však vyskytl hardwarový problém. Naprogramovat mikroprocesor v tomto modulu je velmi obtížné. I při správném nastavení pomocí jumperu na programovací svorkovnici PŘJ většinou dojde k nahodilému stavu a PŘJ přestane komunikovat po sériové lince a dojde buď k nesprávnému naprogramování mikroprocesoru, nebo se mikroprocesor nepodaří naprogramovat vůbec. Je proto nutné použít externí programátor pro naprogramování mikroprocesoru.

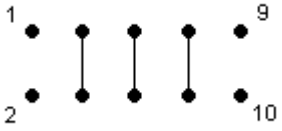
2.3.1 Konektory

1. Pro připojení sériové linky je standardní CANNON 9 typu samice (obsahuje pouze díry pro zasouvací kolíky od samčího konektoru).
2. Pro připojení napájení konektor kruhového průřezu o velikosti 5mm. U tohoto napájecího konektoru musíme dát pozor, aby na zdroji byla nakreslena tato schematická značka (*Obr.11*). Plošný spoj je totiž napájen stejnosměrným napětím. Pokud by se na zdroji nacházela značka s otočenými polaritami, mohlo by při dlouhodobějším přepólování dojít k destrukci celého plošného spoje, nebo alespoň některých součástí. Při krátkodobějším přepólování by zřejmě plošný spoj vydržel bez újmy.

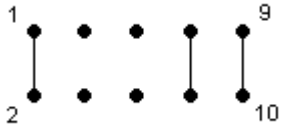
Obr.11 – Polarita napájecího zdroje

2.3.2 Svorkovnice

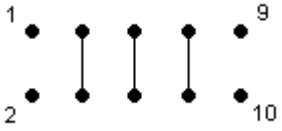
1. X4 – Signály GND (uzemnění), TxD (signál jdoucí po sériové lince z mikročipu do PC), RxD (signál jdoucí po sér.lince z PC do mikročipu).
2. X5 – Signály GND, +5V (napájecí napětí pro většinu modulů),
3. +Ucc (velikost přibližně 12V pro napájení H můstků).
4. SV3 a SV6 – Vedou k Ovládání čidel.
5. SV1 a SV2 – Vedou k H můstkům.
6. JP2 – Tato svorkovnice slouží k nastavování chodu / programováni procesoru. Jak nastavit příslušnou svorkovnici je patrné z následujících obrázků.
   Tato svorkovnice se nastavuje pomocí standardního jumperu používaného u PC.



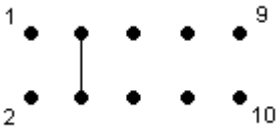
Obr.12 – Dallas programování



Obr.13 – Atmel programování



Obr.14 – Dallas chod



Obr.15 – Atmel chod

Při programování čipu Dallas je ještě nutné držet tlačítko S2.

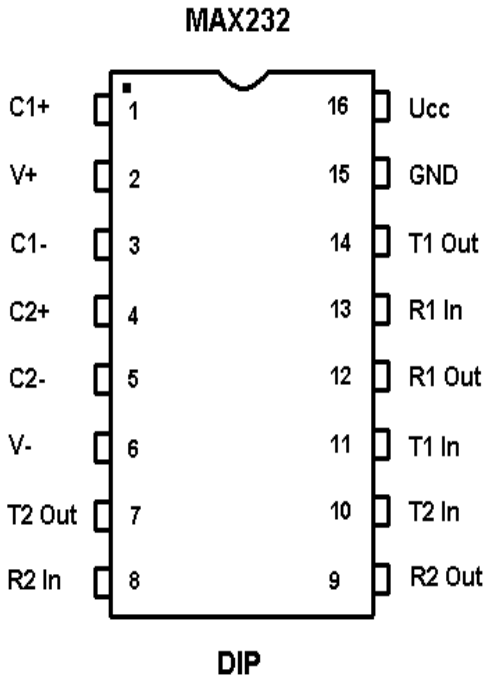
S takto nastavenou svorkovnicí lze pracovat pouze tehdy, je-li námi používaný kabel pro sériovou komunikaci plně hardwarový. Pokud kabel plně hardwarový není, musíme použít jiné nastavení pomocí jumperů, popř. použít příslušnou kombinaci tlačítek. Např. pokud používáme mikročip ATMEL a chceme ho přepnout do programovacího režimu, musíme nejprve stisknout tlačítko SV1, při stálém stisku SV1 přidat ještě tlačítko SV2, poté SV1 uvolnit a nakonec uvolnit i tlačítko SV2. Takto se mikročip přepne do programovacího režimu.

2.3.3 Tlačítka

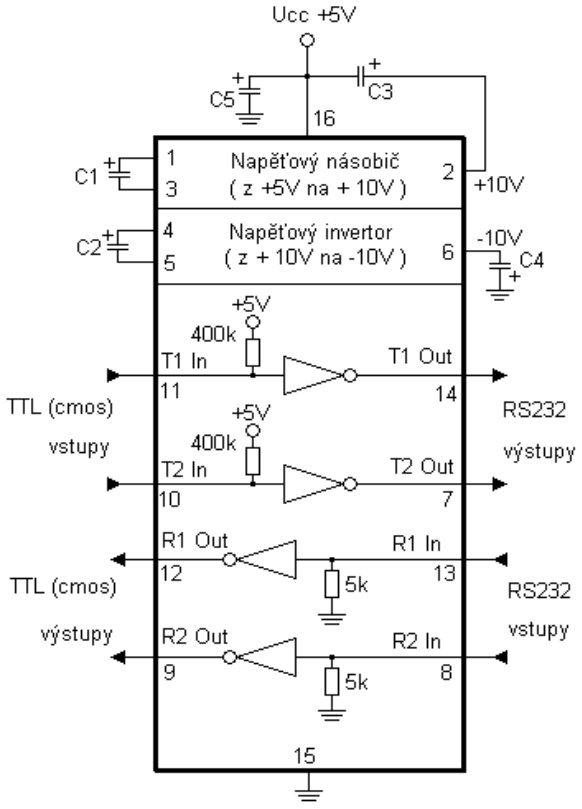
1) S1 – Toto tlačítko slouží jako RESET. Tzn. po stisku tohoto tlačítka dojde k resetu uP pomocí pinu RST.
2) S2 – Toto tlačítko slouží jako programovací tlačítko.

2.3.4 MAX232

Pochází od firmy Maxim Integrated Circuits. Jedná se o obousměrný konvertor RS-232 <=> TTL napájený ze zdroje +5V. Obsahuje 2 konvertory RS-232 → TTL (CMOS) a 2 TTL (CMOS) → RS-232. Oba druhy invertují signál, pro dosažení požadovaného výstupu. Obvod má ke své činnosti zabudovaný napěťový násobič a invertor. Násobič používá kondenzátor C1 (obr.17) ke znásobení napětí +5V na +10V na kondenzátoru C3 (výstup V+) a invertor C2 k inverzi napětí +10V na -10V na kondenzátoru C4 (výstup V-). Jejich hodnota je 1 uF. Obvod má přenosovou rychlost okolo 120kb/s.



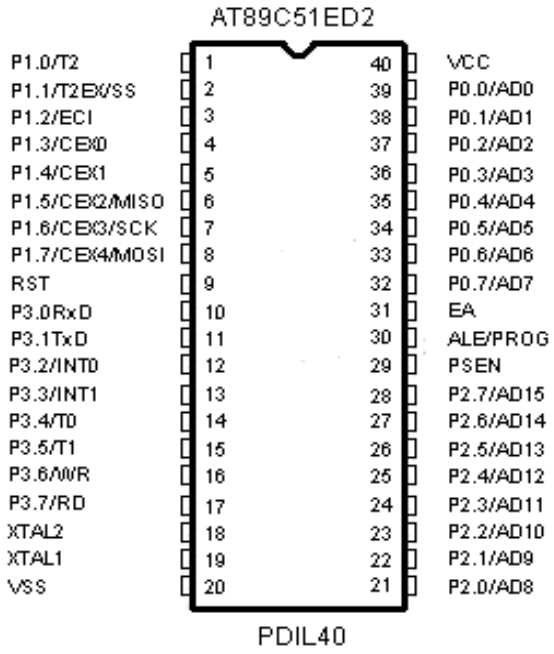
Obr.16 – Zapojení PINů MAX232



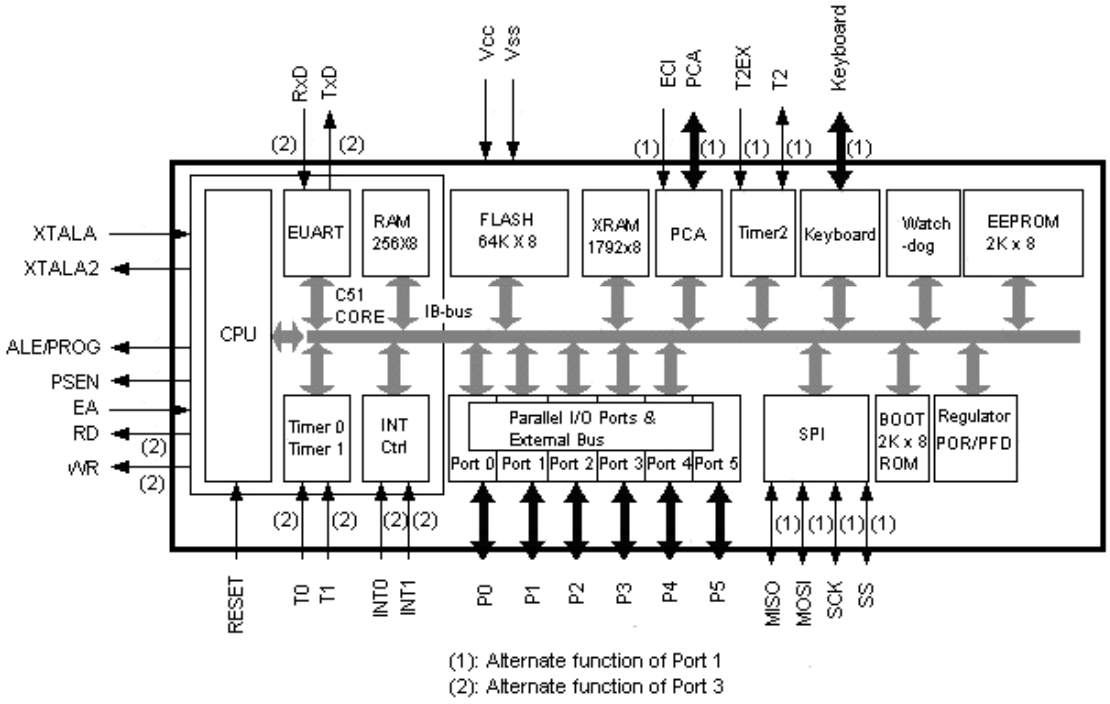
Obr.17 – Aplikační schéma MAX232

2.3.5 ATMEL: (konkrétní označení AT89C51ED2)

- 8 bitový procesor s 16 bitovou sběrnicí
- Procesor programově kompatibilní s instrukční sadou 8051
- 256 Bytů vnitřní paměti
- 64 kBytů vnitřní Flash paměti na program / data
- 2048 Bytů EEPROM blok pro ukládání dat do paměti
- Obsahuje čtyři vstupně / výstupní 8 bitové porty
- Tři 16 bitové čítače / časovače
- Devět přerušení se čtyřmi úrovněmi priority
- Integrated Power Monitor dohlíží na vnitřní napájení
- 16-bit Programmable Counter Array (programovatelný čítač pole)
- AT89C51ED2 je modernější alternativa AT89C51
- Použitý procesor je v pouzdře typu DIL
- Obsahuje 40 vstupně / výstupních pinů
- Nástupce staršího mikroprocesoru AT89C51RD2



Obr.18 – Zapojení PINů



Obr.19 – Blokový diagram AT89C51ED2

Registry funkcí ATMELu

- Střadač (accumulator) – ACC, A

  Základní pracovní registr ALU (aritmeticko logická jednotka). Vždy obsahuje jeden operand aritmetické nebo logické operace. Ukládá se do něj výsledek této operace. Střadač je součástí paměti RAM na adrese 0Eh. Umožňuje kromě normálních operací i bitové operace. Je přístupný jak běžnými instrukcemi (označován A), tak i pomocí přímé adresy (ACC).

- B registr – B

  Pracovní registr. Obsahuje jeden operand, který se využívá při operacích dělení a násobení. Druhý operand je umístěn ve střadači. Spolu se střadačem obsahují výsledek dělení a násobení. Lze ho samozřejmě také použít jako standardní univerzální registr.

- Stavový registr PSW(Program Státu Word)

  PSW obsahuje v každém okamžiku stav procesoru a výsledek výkonu předcházející instrukce. Skládá se z 8-mi bitů. 7 z nich je významových .

Tab.1 – Rozložení příznaků v PSW

b7	b6	b5	b4	b3	b2	b1	b0	Bit
C	AC	F0	RS1	RS0	OV	---	P	Adresa RAM
D7	D6	D5	D4	D3	D2	D1	D0	Bitová adresa

- C – Přenos. Nastavuje se při přenosu z b7 do b8 a při některých instrukcích porovnání.
- AC – Pomocný přenos. Je nastaven, dojde-li při sčítání k přenosu mezi b3 a b4.
- F0 – Uživatelský příznak. Může být libovolně využíván k identifikaci nějaké události (např. přetečení při výpočtu v aritmetice).
- RS1, RS0 – Určují banku. Její registry R0 až R7 pak budou používány.
- OV – Příznak přečtení. Indikuje přečtení při sčítaní (odčítaní).
- P – Příznak parity.Indikuje lichou paritu střadače. Při lichém počtu jedniček potom P=1.

- Ukazatel zásobníku – SP (Stack Pointer)

  Adresový 8bitový registr interní paměti. Ukazuje na vrchol zásobníku. Je zde uložena návratová adresa pro volání funkcí CALL, PUSH a POP. SP je inkrementován při volání funkcí PSUH a CALL a dekrementován po vykonání instrukcí POP a RET. Po inicializaci (resetu) je SP na adrese 07h což znamená, že ukazuje na adresnou buňku 08h. Může být umístěn kdekoli v paměti RAM.

- Ukazatele dat – DPTR (Data Pointer)

  Dvoubytový adresný registr složený ze dvou 8bitových registrů DPH a DPL. Oba registry lze využívat jako paměťová místa. Jeho funkcí je ukazovat na paměťovou buňku v datové paměti v rozsahu 0000h až FFFFh.

- Čítač instrukcí – PC (Program Counter)

  16bitový čítač instrukcí, který není přímo programově přístupný.

- VV porty (Ports) – P0,P1,P2 atd.

  Vstupně / výstupní porty jsou reprezentovány jako registry se stejným označením.

- Registr sériového kanálu – SBUF (Serial Data Buffer)

  Registr sériového kanálu. Ve skutečnosti to jsou dva oddělené registry. Jeden je vysílací a druhý přijímací. Jsou-li data přesunuta do registru SBUF, procházejí vysílacím registrem. Inicializace přenosu se provádí okamžitě. Přicházející data procházejí pouze přes tento registr a musí být z něho přečteny.

Čítače / časovače

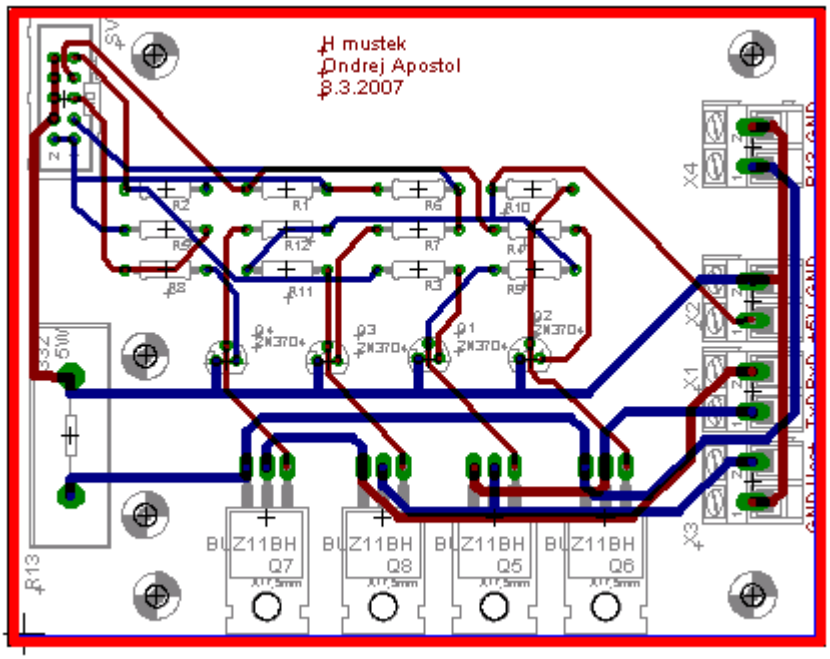
Jejich obsah je přístupný pomocí paměťově mapovaných registrů TH a TL, které určují vyšší a nižší slabiku (8 bitů) příslušného čítače. Hodinový synchronní signál může být určen z oscilátoru procesoru nebo z vnějšího zdroje přivedeného na vývody procesoru T0 a T1. Je-li zdrojem signálu vnitřní oscilátor procesoru, potom je ve funkci časovače a přičítá 1 za každý strojový cyklus, který je tvořen 12 periodami oscilátoru.
Ve funkci čítače vnějších událostí se obsah příslušného čítače zvýší o 1 po přechodu signálu Tn z 1→0. Protože jištění změny na vstupech Tn trvá dva strojové cykly (24 period na oscilátoru), je maximální čítaný kmitočet vnějšího signálu 1/24 kmitočtu oscilátoru mikropočítače. Logická úroveň čítaného signálu musí zůstat nezměněna alespoň 1 celý strojový cyklus. Konfiguraci čítače / časovače 0 a 1 zajišťujeme naprogramováním registru TMOD. Vlastní čítače se programově spouští nebo zastavují nastavením nebo vynulováním bitu TRn v registru TCON.

- TMOD – Registr módů čítačů / časovačů se skládá ze dvou čtveřic bitů, příslušejících každému ze dvou čítačů / časovačů.
- TCON – Registr řízení čítače / časovače se skládá ze čtyř bitů příslušejících oběma časovačům a čtyř bitů patřících vstupům vnějšího přerušení.

Přerušení

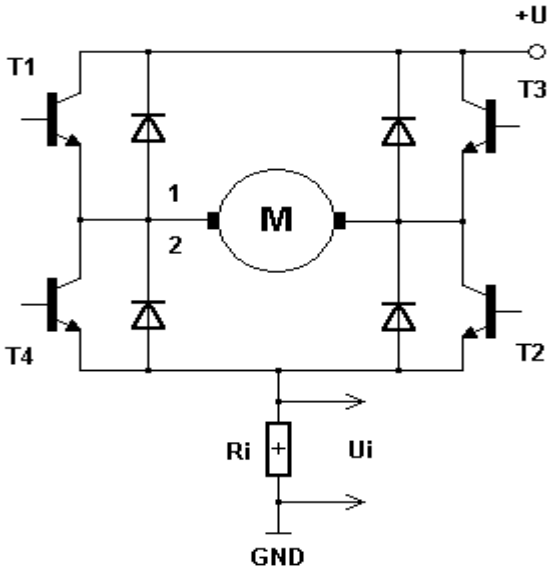
Je určeno pro snazší komunikaci s vnějšími periferiemi. Vnější přerušení INT0 a INT1 mohou být vyvolána buď logickou úrovní (log.0) nebo změnou logické úrovně (sestupnou hranou 1→0). Vznikne-li vnější přerušení, je nastaven příslušní příznak IE0, IE1, který je obvodově automaticky vynulován při volání obslužného podprogramu. Přerušení od časovače 0 a 1 se vyvolávají nastavením příznaků TF0 a TF1, které indikují přečtení příslušného čítače. Vyvolá-li se přerušení od časovače, potom odpovídající znak TFn je vynulován při přechodu do obslužného podprogramu. Přerušení od sériového kanálu se generuje logickým součtem příznaků RI a TI. Tyto dva znaky nejsou automaticky obvodově nulovány při přechodu do obslužného podprogramu, aby uživatel mohl zjistit, zda bylo přerušení generováno příznakem RI (příjem) nebo TI (vysílání). V programu je tedy nutno příznaky nulovat programově. V obslužném podprogramu pro RS232 tak zároveň programově rozhodujeme o tom, která žádost má vyšší prioritu a bude tak zpracována dříve. Každý ze zdrojů přerušení je možné individuálně povolit nebo zakázat nastavením nebo vynulováním příslušného bitu v registru speciálních funkcí IE.

2.4 H můstek



Obr.20 – Návrh desky plošného spoje

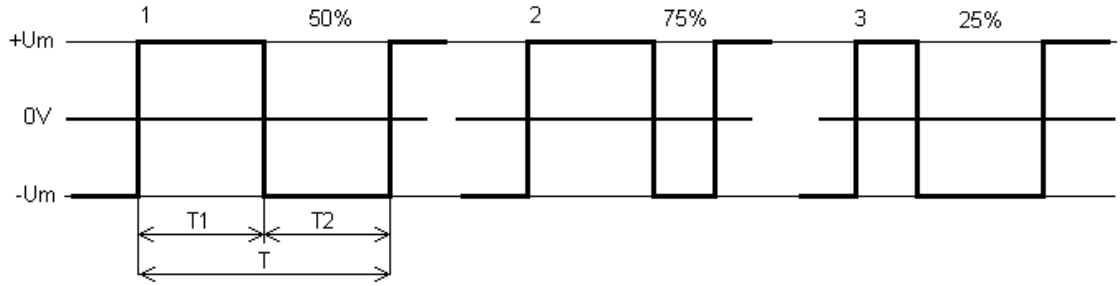
H můstek (jeho název vychází z charakteristického uspořádání tranzistorových spínačů) je zařízení, které nám umožňuje plynulou regulaci napětí na kotvě motorku, reverzaci či jeho brzdění. Tzn. že pomocí H můstku můžeme řídit rychlost / otáčky motorku a směr jeho otáčení. Otáčky stejnosměrného motorku jsou úměrné napájecímu napětí a zatížení. Rychlost tedy můžeme řídit změnou napětí. Protože proud vstupující do motoru je poměrně velký, nelze použít lineární regulaci. Proto se používá PWM (vizte. dále PWM). Při realizaci H můstku jsou v zásadě použity čtyři bipolární tranzistory nebo tranzistory typu MOSFET či IBGT. Z mého návrhu plošné desky je vidět, že tranzistory Q1, Q2, Q3 a Q4 slouží jako spínací tranzistory pro výkonové tranzistory Q5, Q6, Q7 a Q8. Tyto tranzistory slouží k řízení chodu motorku. Všechny odpory mimo odporu R13 slouží jako bázové děliče pro tranzistory. Odpor R13 je výkonový odpor sloužící k měření velikosti proudu procházejícího H můstkem.



Obr.21 – Obecné schéma zapojeni h můstku

Při sepnutých tranzistorech T1 a T2 polarita napětí na motorku odpovídá polaritě v řádku 1. Při sepnutých tranzistorech T3 a T4 je polarita napětí na kotvě opačná (řádek 2). Nikdy nesmí být sepnuty dva tranzistory ve stejné větvi jako T1 a T4 nebo T2 a T3. Malý odpor Ri slouží ke snímání velikosti proudu procházejícího můstkem pro účely proudové zpětné vazby, nebo v jednodušším případě alespoň nadproudové ochrany výkonových tranzistorů.

2.4.1 Řízení můstku dvěma průběhy v protifázy

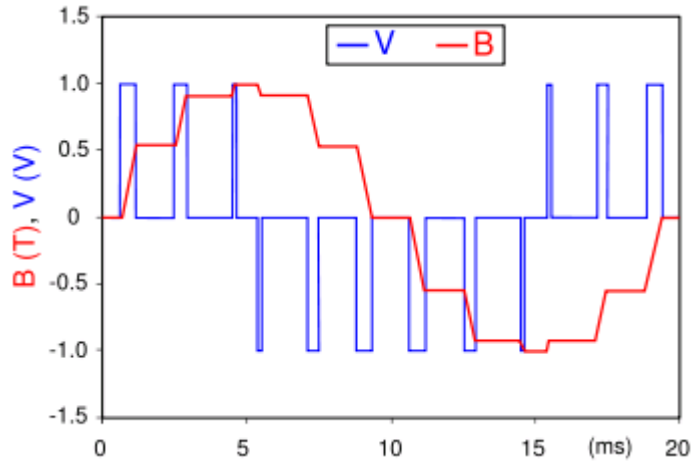


Obr.22 – Řízení můstku dvěma průběhy v protifázi

Během času T1 jsou sepnuty tranzistory T1 a T2 a po dobu T2 tranzistory T3 a T4. Označíme-li napětí o polaritě odpovídající řádku 1 na obr.13 jako kladné, dostaneme při různých poměrech T1 a T2 průběhy na obr.15. Jelikož platí T1 + T2 = T lze pro střední hodnotu napětí přiváděného na kotvu motoru psát UaS = (T1 – T2) Um /T. Při rovnosti T1 = T2 je střední hodnota napětí přivedeného na kotvu motoru nulová a motor tudíž stojí. Při nerovnosti je podle znaménka buď kladná nebo záporná, přičemž její velikost lze plynule regulovat, a motorek se otáčí buď jedním nebo druhým směrem.

2.4.2 PWM (Pulse Width Modulation) aneb pulsně šířková modulace

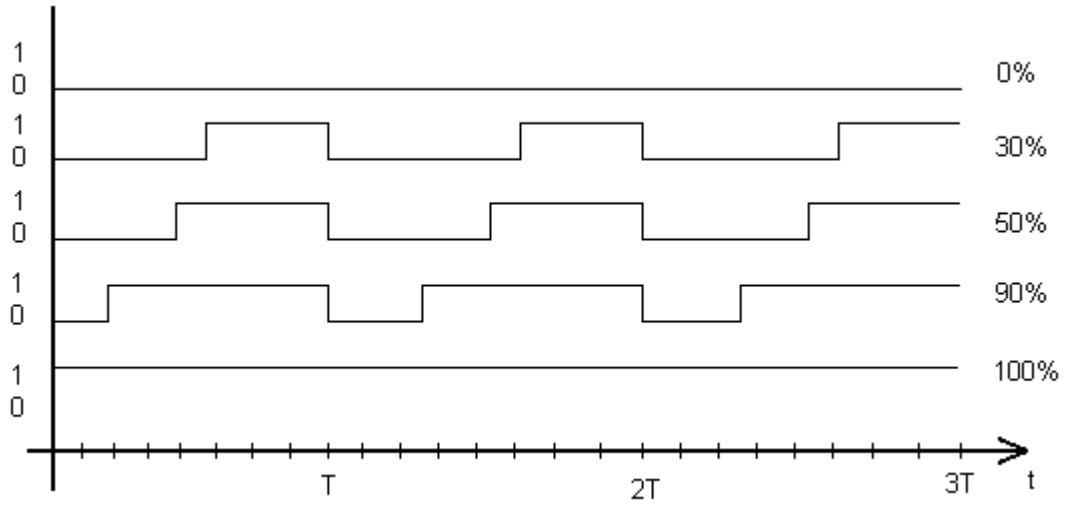
Obecně se jedná o modulaci pomocí šířky pulsu. Změnou poměru doby trvání napěťového pulsu ku periodě se mění střední hodnota napětí.



Modrý pulsující signál U[V] je pomocí PWM přetransformován na červený, spojitý, přibližně sinusový signál. Hladkost křivky lze řídit pomocí šířky pulsu a jejich množstvím.

Obr.23 Princip PWM
Použitá literatura [6]

Konkrétně jde o signál s konstantní periodou T, kde se mění střída napětí (tj. poměr délky impulzu ku délce mezery uvažovaný v jedné periodě). Střída se uvádí někdy jako poměr (1:1,2:1,1:5 atd.), kdy je nutné uvést které číslo představuje impulz a které mezeru. Někdy se střída vyjadřuje procentuálně (100%,50%,0.1% atd.), kde 100% představuje ideální poměr 1:0, 50% poměr 1:1 atd. Poměr délky impulzu ku délce mezery bývá v zahraniční literatuře nazýván Duty Cycle.



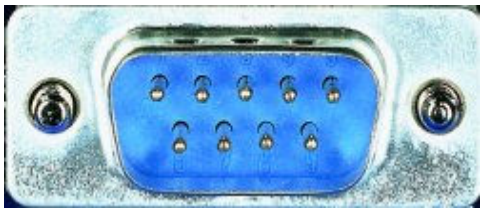
Obr.24 – Časový průběh signálu s různou střídou

U ovládání stejnosměrného motorku pomocí PWM spočívá její princip v rychlém spínání a vypínání napájecího napětí. Díky setrvačnosti motorku a dostatečně vysoké frekvenci spínání, rotor nestačí tyto změny sledovat. Motorek se chová jako kdyby byl napájen napětím o velikosti střední (průměrné) hodnoty, která je dána poměrem doby zapnutí a vypnutí.

RS232 – Sériová linka

Je rozhraní pro přenos informací vytvořené původně pro komunikaci dvou zařízení do vzdálenosti 20m. Pro větší odolnost proti rušení je informace po propojovacích vodičích přenášena větším napětím, než je standardních 5 V. Přenos informací probíhá asynchronně, pomocí pevně nastavené přenosové rychlosti a synchronizace sestupnou hranou startovacího impulzu. Pro realizaci konektorů je možné více variant.

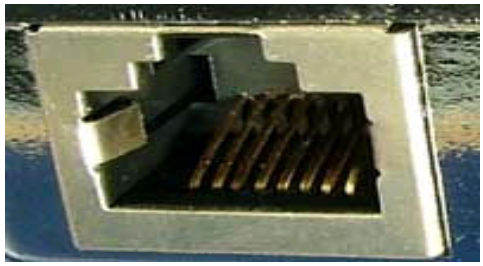
Nejpoužívanější z nich jsou CANNON 9 (*Obr.25.1*), CANNON 25 (*Obr.25.2*) a RJ45 (*Obr.25.3.1 a Obr.25.3.2*). U našeho přípravku je použit konektor CANNON9.



Obr.25.1

Obr.25.2

Obr.25.3.1



Obr.25.3.2

Tab.2 – Jednotlivé žíly kabelu sériové linky

Vývod	Název	Směr	Původní význam
1	RLSD (DCD)	Vstup	Detektor nosného signálu
2	RxD	Vstup	Přijímaná data
3	TxD	Výstup	Vysílaná data
4	DTR	Výstup	Pohotovost koncového zařízení
5	GND	---	Signálová zem
6	DSR	Vstup	Pohotovost ukončovacího zařízení
7	RTS	Výstup	Výzva k vysílání
8	CTS	Vstup	Pohotovost k vysílání
9	RI nebo RING	Vstup	Indikátor volání

2.5.1 Základní parametry

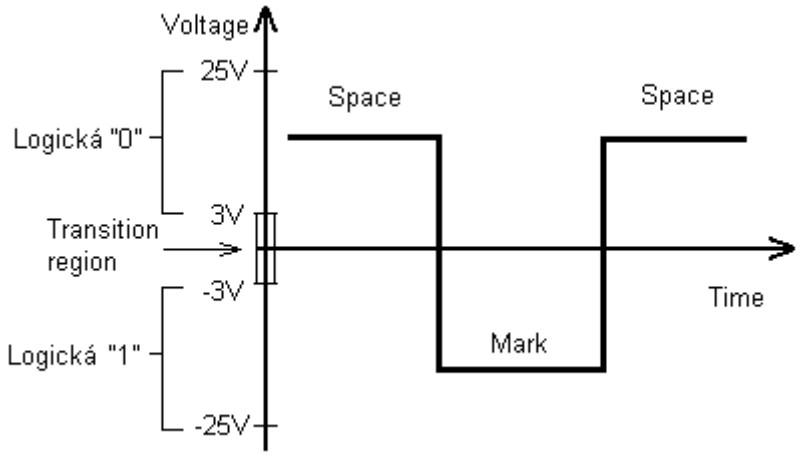
RS 232 používá dvě napěťové úrovně. Logickou 1 a 0. Log.1 je někdy označována jako marking state nebo také klidový stav, Log.0 se přezdívá space state. Log.1 je indikována zápornou úrovní, zatímco logická 0 je přenášena kladnou úrovní výstupních vodičů.

Nejběžněji se pro generování napětí používá napěťový zdvojovač z 5 V a invertor. Logické úrovně jsou potom přenášeny napětím +10 V pro log.0 a –10 V pro log.1.

Tab.3 – Logické úrovně

Datové signály		
Úroveň	**Vysílač**	**Přijímač**
Log. L	+5V až +15V	+3V až +25V
Log. H	-5V až -15V	-3V až -25V
Nedefinovaný	-3V až 3V	

Řídící signály		
Signál	**Driver**	**Terminátor**
„Off“	-5V až -15V	-3V až -25V
„On“	5V až 15V	3V až 25V



Obr.26 Povolené napěťové úrovně

Pro komunikaci po sériové lince nemusí být nutně všechny piny obsazeny.U naší Programovatelné řídící jednotky nejsou využity piny DSR, RTS a RI. Jedná se o vstupy, které slouží k řízení procesu a nebylo nezbytně nutné je použít. Protože je sériová linka plně duplexní, umožňuje přenášení dat oběma směry současně.

3. Software

Software lze v něm případě rozdělit na dvě části. Jedna část se zabývá ovládacím softwarem, který se nachází v PC a druhá část programem, který je přímo v uP.

3.1 Software v PC

Tento software byl vytvořen pomocí vývojového prostředí Delphi 6, který vyvinula firma Borland. Je určen pro tvorbu aplikací v platformě MS Windows. Používaný programovací jazyk v tomto vývojové prostředí je Object Pascal (jedná se o objektovou nástavbu standardního Pascalu). Je zde implementován systém RAD (Rapid Application Development), umožňující vizuální návrh grafického uživatelského rozhraní, na jehož základě je přímo tvořena kostra zdrojového kódu. Díky tomu je výrazně urychlen vývojový cyklus. Programování v Delphi je převážně založeno na používání implementovaných komponent. Komponenta je balíček funkcí (program), který dle svého určení vykonává určitou činnost (komunikuje s databázemi, zprostředkovává FTP přenos, umožňuje zobrazovat text,obrázky či přehrávání medií). Velkou výhodou Delphi před konkurenčními programy jsou knihovny komponent, které jsou jeho součástí (VCL, CLX atd.). Tyto komponenty nám výrazně ulehčují tvorbu aplikací. Další komponenty je možné stáhnout z internetu či sehnat na různých CD. Některé tyto knihovny jsou volně šiřitelné (free), ostatní jsou zpoplatněné. Delphi samozřejmě umožňuje tvorbu vlastních komponent.

3.1.1 Zvolená knihovna komponent sériové komunikace

Sériovou komunikaci lze jednoduše realizovat s využitím naprogramovaných komponent sériové komunikace pro VCL. Mnohé z těchto komponent jsou volně dostupné, především pro vývojové prostředí Delphi [31], ale i pro C++ Builder. Snad nejrozsáhlejším volně dostupným produktem je donedávna komerčně distribuovaný Async PRO od firmy Turbopower. Další možností jsou balíčky ComPort Library, Varian Async32, nebo velmi podobný TMS Async 32.

Po krátkém testování výše uvedených produktů jsem se rozhodl použít ComPort Library. Knihovna je v současné době dostupná pro Delphi 3, 4, 5, 6 a C++ Builder 3, 4, 5, 6 na OS Windows 95, 98, NT 4, 2000. Autorem je Dejan Crnila, podporu pro C++ Builder provedl Paul Doland.

Vlastnosti:

• platforma: Windows NT 4.0, Windows 2000, Windows 95, Windows 98
• podporované programovací jazyky: Delphi 3, 4, 5, 6 a C++ Builder 3, 4, 5, 6
• synchronní /asynchronní operace (čtení/zápis)
• detailní možnosti řízení komunikace
• timeout pro operace čtení/zápis
• použití vícevláknového programování pro monitorování událostí portu
• vestavěná aplikace terminálu
• volně dostupný zdrojový kód (asi 7000 řádků)
• kontextový help

Knihovna obsahuje následující komponenty:

- TComPort základní komponenta pro úplnou sériovou komunikaci
- TComDataPacket umožňuje přijímání dat v paketech
- TComComboBox combo box pro nastavení sériového portu
- TComRadioGroup radio group pro nastavení sériového portu
- TComLed umožňuje zobrazení stavu signálů sériové linky
- TComTerminal terminál VT52, VT100 a ANSI

3.1.1.1 Komunikace v paketech

Jestliže je aplikace připojena k sériovému zařízení typu datalogger, které neustále posílá data do PC, je užitečné posílat data v paketech. Paket je řetězec znaků, obvykle konstantní délky, obsahující start a někdy i stop bit. Aplikace může analyzovat přicházející data uvnitř události OnRxChar, ale mnohem jednodušší je použití komponenty TComDataPacket, která provádí analýzu příchozího řetězce automaticky. Komponenta TCustomComPort umožňuje připojit více než jednu komponentu TComDataPacket, takže aplikace může jednoduše přijímat pakety rozdílné délky a typu.

Když je paket vytvořen (zformován), je vyvolána událost OnPacket s řetězcem paketu jako parametrem. Všechna data, která jsou odřezána během tvorby paketu je možno získat událostí OnDiscard.

Jestliže je komponenta TComDataPacket spojena s TCustomComPort, událost OnRxChar komponenty TCustomComPort není spouštěna, takže aplikace využívá událost OnRxBuf jestliže požaduje přijmout neanalyzovaná data.

3.1.1.2 Zápis

Operace zápisu se provedou velmi jednoduše voláním příslušné metody. Obsahuje čtyři metody, které provádí zápis dat.

Metody zápisu:

- Write Zápis netypové proměnné do výstupního bufferu.
- WriteAsync Zápis netypové proměnné do výstupního bufferu v asynchronním módu.
- WriteStr Zápis proměnné typu řetězec do výstupního bufferu.
- WriteStrAsync Zápis proměnné typu řetězec do výstupního bufferu v asynchronním módu.

3.1.1.3 Čtení

Čtení ze vstupního bufferu může být provedeno dvojím způsobem. Obvykle aplikace volá nějakou metodu čtení uvnitř události OnRxChar, která je vyvolána jestliže znak nebo znaky přijdou do vstupního bufferu. Metoda čtení tedy kontroluje vstupní buffer a vrací jeho obsah okamžitě kdy už je počet bajtů ve vstupním bufferu známý. Aplikace může také volat metodu čtení mimo událost OnRxChar.

Jestliže je komponenta připojena k jiné komponentě, jenž vyžaduje vstupní data, jako např. TComDataPacket nebo TCustomComTerminal, událost OnRxChar není volána. Místo ní komponenta volá událost OnRxBuf. Aplikace nemůže číst data ze vstupního bufferu uvnitř události OnRxBuf, protože už byla přečtena a přesunuta jinam. Data jsou komponentou automaticky umístěna do parametru Buffer události OnRxBuf. Jestli je volána událost OnRxChar nebo OnRxBuf, může být ověřeno vlastností TriggersOnRxChar.

Metody čtení:

- Read Čtení ze vstupního bufferu do netypové proměnné.
- ReadAsync Čtení ze vstupního bufferu do netypové proměnné v asynchronním módu.
- ReadStr Čtení ze vstupního bufferu do proměnné typu řetězec.
- ReadStrAsync Čtení ze vstupního bufferu do proměnné typu řetězec v asynchronním módu.

3.1.1.4 Vyčištění vstupního a výstupního bufferu

Aplikace může vyčistit vstupní i výstupní buffer použitím metody ClearBuffer. Před použitím se ujistěte, že neprobíhají asynchronní operace když aplikace volá ClearBuffer, protože to může způsobit nepředvídatelné problémy.

3.1.1.5 Otevření a uzavření portu.

Předtím, než může být většina metod komponenty TCustomComPort úspěšně volána, sériový port musí být otevřen. Za tímto účelem existují dvě možnosti. Aplikace může volat metodu Open nebo nastavit vlastnost Connected na True. Pro uzavření lze volat metodu Close nebo nastavit vlastnost Connected na False.

3.1.2 Komunikační protokol

Komunikační protokol je program, který nám umožňuje komunikaci mezi dvěma zařízeními. Může se jednat o komunikaci mezi dvěma PC, PC a nějaké zařízení nebo mezi dvěma zařízeními. V našem případě se jedná o komunikaci mezi PC a zařízením (PŘJ). Jelikož se PŘJ nachází mikroprocesor, je vhodné použít buď binární komunikaci, nebo ASII komunikaci. Při binární komunikaci se po spojovací lince posílají 1 a 0, kdežto u ASII komunikaci se po lince posílají znaky, které jsou standardně obsaženy v ASII tabulce. Může se jednat o velká či malá písmenka, číslovky, různé znaky jako jsou %, #, ( ) atd..

U komunikačního protokolu jsem se rozhodl použít ASCII komunikaci, která je sice o něco náročnější pro pochopení a naprogramování v PC, ale jednodušší pro dekódování v uP. Samotná komunikace probíhá v paketech. Každý paket má přesně definovaný konec a začátek.Tyto dva znaky se nazývají prefix (začátek) a sufix (konec).

Prefix jsem definoval jako znak Escape a sufix jako znak Enter. Dále se zde posílají další tři znaky.

Jako první znak se tedy posílá prefix. Následující znak určuje směr otáčení obou motorků (obou motorků naráz, abych ušetřil jeden znak). Jaký znak se aktuálně posílá je vidět z tabulky (*Tab.5*). Další dva znaky určují rychlosti otáčení motorků. Pokud se bude jednat o rychlost nula, bude posílán znak „a“. Při každé změně rychlosti o jedna se zvýší i posílaný znak o 1. Například při rychlosti motorku 3 se pošle znak „d“ . Poslední konečný znak je sufix. Takže pokud například chceme, aby se motorek 1 otáčel vpravo a měl rychlost 3 a motorek 2 otáčel vlevo s rychlostí 8, pošle se po sériové lince paket s těmito daty : 1B 62 64 69 0D v hexadecimálním vyjádření, ESC b d i ENTER jako znaky. Tato data se posílají z počítače do PŘJ. Pokud chceme zavolat z obslužného programu kódování dat, musíme zavolat funkci Encode. Tato funkce dokáže převést aktuální data na požadovaný řetězec typu String.

Komunikační protokol samozřejmě musí i umět dekódovat data, která přichází z PŘJ. Pro dekódování dat přicházejících ze sériové linky slouží funkce Decode. Tato funkce v sobě obsahuje kontrolu, zda je paket platný a není příliš krátký. Pokud by byl např. paket příliš krátký, mohlo by dojít ke špatnému přiřazení. Směr otáčení by mohl být přiřazen otáčkám a otáčky by nebyly definovány vůbec. Mohlo by tak dojít k nestabilitě obslužného programu, nebo k jeho nesprávné funkci. Dále se zde mohou vyskytnout náhodné rušivé signály, které by také mohli ovlivnit správnou funkčnost ovládacího panelu. Proto funkce Decode kontroluje, zda je velikost paketu skutečně 8 a nejedná se o chybný paket.

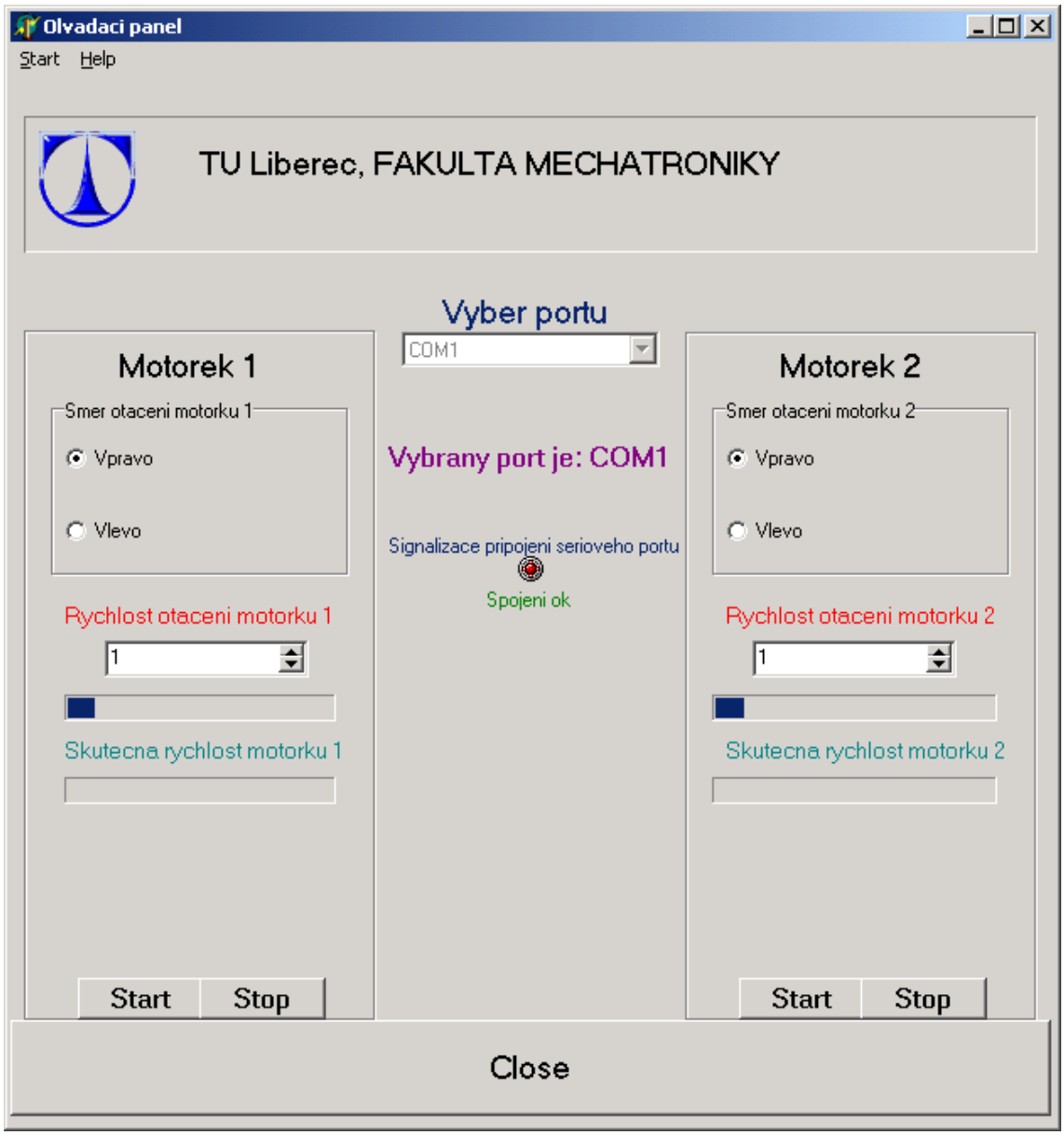
Tab.5 – Aktuální znak

Motorek 1	Motorek 2	Znak
Pravá	Pravá	a
Pravá	Levá	b
Levá	Pravá	c
Levá	Levá	d

Mnou vytvořený komunikační protokol je poměrně jednoduchý a měl by být široce implementovatelný i pro jiná zařízení. Je však nutno brát v potaz, že posílaná data jsou v ASCII formátu, nikoli v binárním a ne každé zařízení je schopno správně dekódovat data typu String. Některá zařízení totiž umožňují pouze binární komunikaci.

3.1.3 Ovládací panel

Ovládací panel je software, které běží přímo v PC. Je stejně jako komunikační protokol napsán v programu Delphi 6. Pro realizaci Ovládacího panelu jsem použil standardních knihoven komponent. Je zde také použita speciální knihovna ComPort Library, která je podrobně popsána výše. Ovládací panel je přímo spojen s Komunikačním protokolem, takže se uživatel při ovládání motorku nemusí obtěžovat posíláním dat, nebo spouštěním jiného programu. Ovládací panel umožňuje uživateli měnit směr otáčení motorku, jeho rychlost a dále vyhodnocuje jakým směrem se motorek otáčí a jakou má aktuální rychlost. Pro komunikaci s motorkem je nejprve nutné vybrat port, ke kterému je zařízení připojeno. Rozsah portů je COM1 až COM6, což by mělo být plně postačující i za předpokladu, že používáme převodník USB→RS232, který většinou definuje COM port od čísla 4 a výše. Toto číslo lze samozřejmě softwarově změnit. Při změně postupujeme takto: pravým tlačítkem klikneme na Tento počítač→Vlastnosti, vybereme záložku Hardware→Správce zařízení→Porty(COM a LPT) vybereme příslušný port a v Nastavení portu→Upřesnit vybereme číslo příslušného portu. Je však nutné dát pozor, abychom nezměnili číslo portu na port, který je již obsazen, mohlo by totiž docházet ke konfliktu zařízení. Pokud proběhne spojení s portem úspěšně, rozsvítí se červená led dioda a objeví nápis Spojení ok. Po výběru portu a jeho potvrzení klikem na tlačítko Start komunikace již můžeme libovolně měnit rychlosti a směry otáčení motorku. Každý motorek má pro ovládání svoje tlačítko Start a Stop. Tlačítkem Start odesíláme požadované změny v rychlosti a směru otáčení a tlačítkem Stop zastavujeme chod motorku. ProgressBar nám pak zobrazuje skutečnou aktuální rychlost a Labely zobrazují skutečný směr otáčení motorku. Tyto data jsou přijímána zpětně z RS232. Pokud tedy změníme rychlost či směr může motorku chvíli trvat, než na tuto změnu zareaguje. A právě po tuto dobu budou odesílaná a přijímaná data rozdílná. Jinak by měli být přibližně stejné. Je zde také přiložen Help, který uživatele seznamuje s podrobnějším postupem při práci s Ovládacím panelem.



Obr.27 Náhled na Ovládací panel

Jak je vidět z obrázku Ovládacího panelu levý sloupec je vyhrazen pro ovládání a snímání otáček motorku 1 a pravý pro motorek 2.

3.2 Programování mikroprocesoru

Pro naprogramování mikroprocesoru se v praxi používají hlavně dva programovací jazyky. Velmi rozšířeným standardem je Assembler. Dále jsou hojně využívány některé z překladačů vyššího programovacího jazyka pro procesory v jazyce C.

3.2.1 Assembler je programovací jazyk, který je velmi podobný strojovému kódu. Je to jazyk nejnižší úrovně a téměř každý procesor má svůj odlišný assembler. Tento jazyk je v základu tvořen dvěma částmi *Direktivami* a *Instrukcemi.*
Direktivy – Nevytvářejí přímo kód programu, ale pouze řídí činnost překladače
Instrukce – Jsou symbolickým zápisem strojových instrukcí procesoru.
Tento programovací jazyk patří mezi nejstarší vůbec, proto bychom zde marně hledali věci běžné v ostatních programovacích jazycích, jako jsou IF, WHILE atd. V tomto jazyku existují pouze instrukce. Jsou zde však ve velké míře používány návěští a různé typy skoků(dlouhé,krátké,podmíněné atd.).

3.2.2 Jazyk C je jazyk určen pro nízkoúrovňové programování. Díky tomu se prosadil hlavně při realizaci programů pro ovládání libovolného hardwaru.
Výhody oproti assembleru

- Jazyk je multiplatformní, tzn. snadnou přenositelnost zdrojových kódů programu.
- Celková přehlednost zdrojových kódů programu a zjednodušení při správě složitějších projektů.
- Dostupnost nástrojů na optimalizaci a validaci výsledného kódu.
- Výrazné urychlení práce pro vývoj aplikací.

Nevýhoda je hlavně ve vyšší náročnosti na paměť dat programu a programu výsledné aplikace. Dále se nejedná o volně šiřitelný program, je tedy nutné zakoupit originální licenci. Oproti tomu, je většina starších vývojových prostředí pro assembler volně šiřitelná. Většinou se jedná o programy, které běží pod operačním systémem MS DOS a není tedy nutné investovat další prostředky.

Implementace pro x51

Procesory řady x51 patří stále mezi velmi oblíbené a díky tomu je na trhu velké množství kompilátorů jazyka C. Mezi nejzákladnější vlastnosti jazyka C je jeho využívání práce se zásobníkem. Používá se pro předávaní parametrů, lokální proměnné, návratové adresy funkcí atd.. Toto je však velký problém pro procesory řady x51, protože ty neobsahují vhodný zásobníkový systém. Implementace jazyka C tedy závisí pouze na kompilátoru a jak se dokázal daný výrobce s tímto problémem vypořádat.

3.2.3 Kompilátory

- **GNU kompilátor SDCC** (Smart Device C Compiler) je vyvíjený skupinou nadšenců. V porovnání s komerčními produkty vede docela obstojně. Svými vlastnostmi výsledného kódu programu se řadí do střední třídy kompilátorů. Chybí spousta vymožeností, které je dána absencí IDE a s tím spojený komfort při vývoji. Rozšířená syntaxe jazyka C používaná u mikroprocesorů řady 8051 je u SDCC a Keil C51 shodná. To umožňuje psát programy, které lze jednoduše přeložit v obou kompilátorech. Při realizaci složitějších projektů, je však třeba počítat se zvýšeným úsilím, ze strany vývojáře k dosažení patřičného výsledku.

- **Keil C51** je odbornou veřejností považován za jeden z nejlepších kompilátorů pro 8051 mikroprocesory. Na celém produktu je vidět několik let intenzivního vývoje, který má za sebou. Nevýhodou však může být, že je oproti SDCC komerční, tzn. pokud ho chceme využívat, musíme si pořídit licenci, která je poměrně drahá. Keil naopak obsahuje mnoho speciálních knihoven a umožňuje optimalizaci pro velké množství rozdílných mikrokontrolérů. Práce v něm je tedy omnoho pohodlnější než v konkurenčním SDCC.

- Samozřejmě existuje mnoho dalších kompilátorů, které lze využít pro naprogramování mikroprocesoru, např. : ANSI C x51 Compiler, ImageCraft ICCAVR, CodeVisionAVR a další.

3.2.4 Realizace programu

Obslužný program byl napsán pomocí kompilátoru Keil a měl by hlavně umět dvě základní věci.

1, Přijímat data ze sériové linky, dekódovat je a podle nich vytvořit pulsně šířkovou modulaci pro ovládání h můstků resp. motorků.

Čtení dat z RS232

```
void seriál_IT (void) interrupt 4
 {
        if  (RI = = 1)                          // indikace naplnění přijímacího registru
        {
        buffer[write_pointer]=SBUF;             // ulož data z SBUFu
        write_pointer++;                        // přičti 1
        if (write_pointer = = buffer_length) write_pointer=0; // pokud se rovnají vynuluj
        }
        RI = 0;                                 // nulování RI
}
```

Po přijetí dat tímto cyklem jsou následně data dekódována a podle směru se nastaví na příslušné výstupní porty procesoru (P1.4 až P1.7 a P2.4 až P2.7) jednička a nuly. Na zbylý jeden port procesoru se přivede PWM, která ovládá rychlost otáčení příslušného motorku. PWM je počítána z času zapnutí a času vypnutí.

$$T_{zap} = \frac{\frac{T1clock}{frekvence}}{2} + \frac{K}{N} \qquad T_{vyp} = \frac{\frac{T1clock}{frekvence}}{2} - \frac{K}{N}$$

T1clock…rychlost krystalu / 12

K…T1clock / 256

N…vychází ze znaku přijatého po RS232

Z času zapnutí a vypnutí se pak vygeneruje aktivní a neaktivní část průběhu. Aktivní částí průběhu se rozumí čas, po který je signál na log.1 a neaktivní log.0. Z těchto dvou průběhů pak vznikne střední hodnota odpovídající požadované hodnotě.

2, Odesílat data z vyhodnocovačů IR čidel o chování motorků po sérové lince zpět do PC. Tato data jsou posílána z PŘJ do PC pomocí sériové linky ve formě znaků. Rychlosti otáčení jsou od 0 do 10. Kde rychlost 0 je znak „A“ a rychlost 10 znak „K“. Tyto rychlosti se počítají na základě impulsů, které generují čidla resp. vyhodnocovače signálu z IR čidel.

Počítání pulsů

```
unsigned char temp_r;                 // deklarace temp_r
static unsigned char red_old;         // deklarace red_old
…
temp_r = P2_1;                        // přiřazení portu P2.1 do temp_r
if (temp_r != red_old) speed_imp++; // pokud se nerovnají přičti 1 do impulzu
red_old = temp_r;                     // nastavení rovnosti red_old a temp_r
```

Vyhodnocení signálů

```
if  (speed_imp == 0 ) posli[1] = 'A';          // pokud počet pulsu=0 posli A
else if  (1<=speed_imp <= 10 ) posli[1] = 'B';  // počet pulsu v rozmezi posli B
...
```

Jak je vidět z přiloženého zdrojového kódu rozhodování o rychlosti otáčení je děláno přes cykly if jednoduchou logikou.

Dále se zasílají data o směru otáčení Směr otáčení vlevo je definován jako znak „A“ a směr vlevo jako „B“. Směry a rychlosti obou motorků jsou zasílány naráz. Pokud se tedy bude motorek 1 otáčet rychlostí 2 vlevo a motorek 2 rychlostí 5 vpravo bude zasílaný paket vypadat takto: CAFB. Tento paket je následně v PC dekódován a rychlosti a směry otáčení jsou zobrazeny na Ovládacím panelu. Data se posílají pomocí registru SBUF. Pokud chceme data odeslat po RS232 stačí pouze přiřadit tyto data např. SBUF=posli.

Závěr

Cílem bakalářské práce bylo realizovat zpětnovazební řízení malého stejnosměrného motorku tzn.realizovat a uvést do funkčního stavu hardware, vyvinout potřebný obslužný software pro PC a naprogramovat příslušný mikroprocesor tak, aby byl schopen ovládat celé zařízení a zároveň obousměrně komunikovat s PC.

Při testování hardwaru sice musely být provedeny drobné úpravy v návrhu desek plošných spojů a vyměněny některé nefunkční nebo špatně fungující součástky, ale v konečné fázi je hardware při připojení všech příslušných modulů plně funkční .

Software v PC je odladěn a plně funkční. Je tedy schopen dekódovat data, zaslat je jako paket po RS232 do PŘJ a umožnit nám tím řízení příslušného hardwaru resp. motorků a měnit jejich směr otáčení a rychlost pomocí PC. Dále je schopen přijmout data z PŘJ, dekódovat je a zobrazit příslušný výsledek o chování motorků.

Celé toto zařízení bych považoval za velmi dobře rozšiřitelné. Je možné s ním realizovat např. automatické řízení jeřábu. Jeřáb by byl osazen senzory na snímání polohy, mikroprocesor by nám tato data zasílal do PC, kde by je příslušný program dekódoval a analyzoval. Software v PC by pak ovládal příslušné motorky tak, aby jeřáb vyzvedl zaměřený objekt a přemístil ho na požadované souřadnice.

Seznam použité literatury

[1] Skalický, P. *Mikroprocesory řady 8051*. Praha: Vydavatelství: BEN – technická litertura, 2003. ISBN 80-86056-39-2

[2] Matoušek, D. *Práce s mikrokontroléry ATMEL*. Praha: Vydavatelství BEN – technická literatura, 2002. ISBN 80-7300-094-6

[3] Hlava, J. *Prostředky automatizovaného řízení II*. Praha: Vydavatelství ČVUT, 2000.

[4] Němeček, S. a kol. *Technické prostředky automatizovaného řízení – část I*. Liberec: Vydavatelství VŠST Liberec,1982.

[5] *Wikipedia. Wikipedia [Internet]. http://en.wikipedia.org/wiki/Electric_motor*

[6] *Wikipedia. Wikipedia[Internet] .http://en.wikipedia.org/wiki/Pulse-width_modulation*

[7] Atmel. *Atmel* .[Interenet]. www.atmel.com/dyn/resources/prod_documents/doc3ae686c15593f.pdf

[8] Texas Instruments. *Texas Instruments* [Internet]. http://focus.ti.com/docs/prod/folders/print/max232e.html

[9] HW group. *HW group* [Internet]. http://www.hw-group.com/support/bcc_examples/ch1com_example_en.html

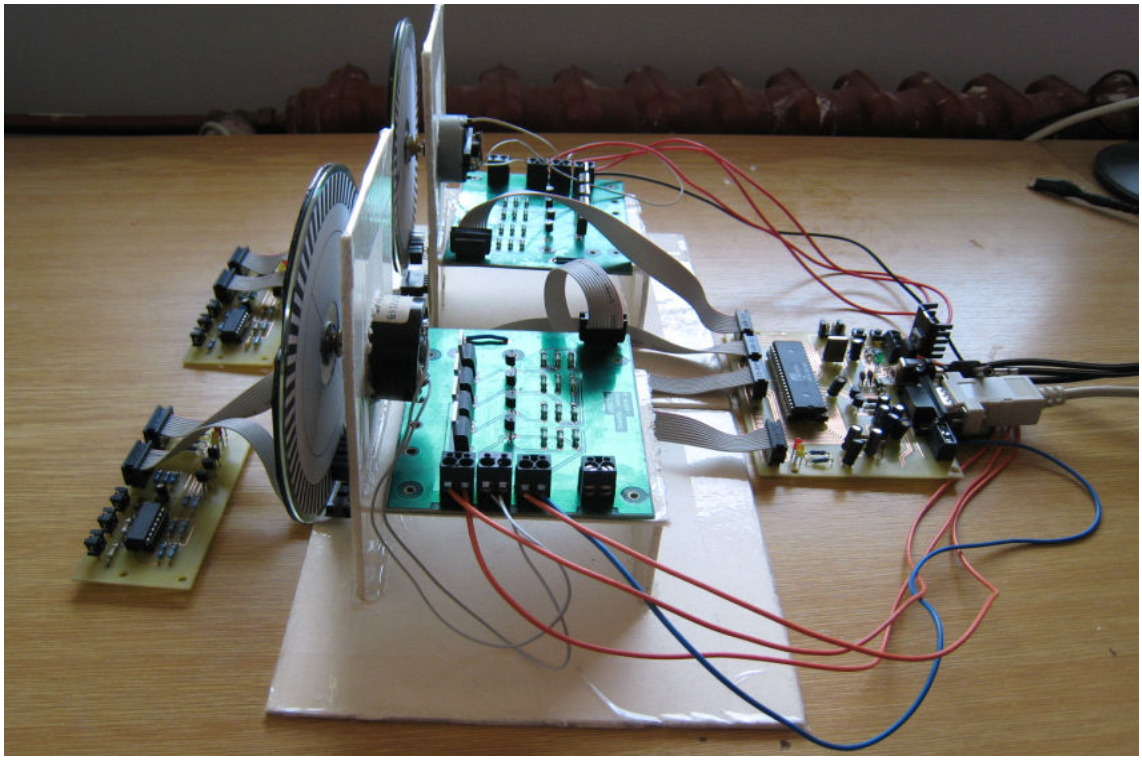
[10] Papouch s.r.o. *Papouch s.r.o.* [Internet]. http://www.papouch.com/shop/scripts/_spinel.asp

[11] Builder. *Builder* [Internet]. http://www.builder.cz/serial3.html

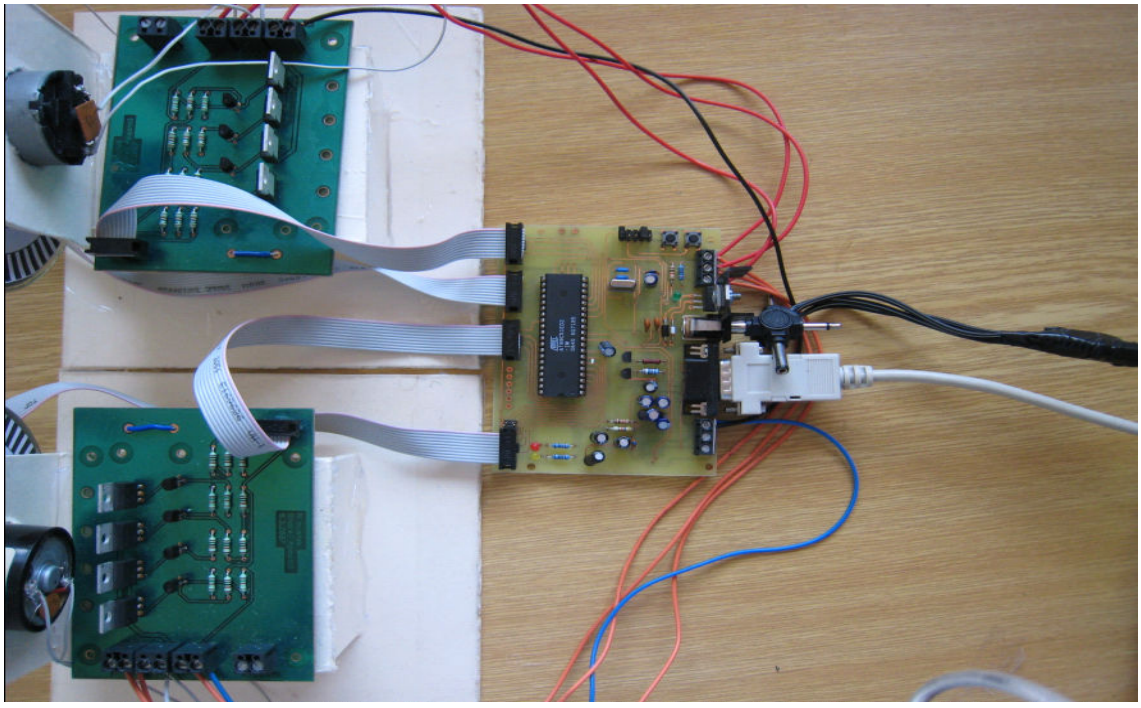
[12] Keil. *Keil* [Internet]. http://www.keil.com/c51/

Příloha

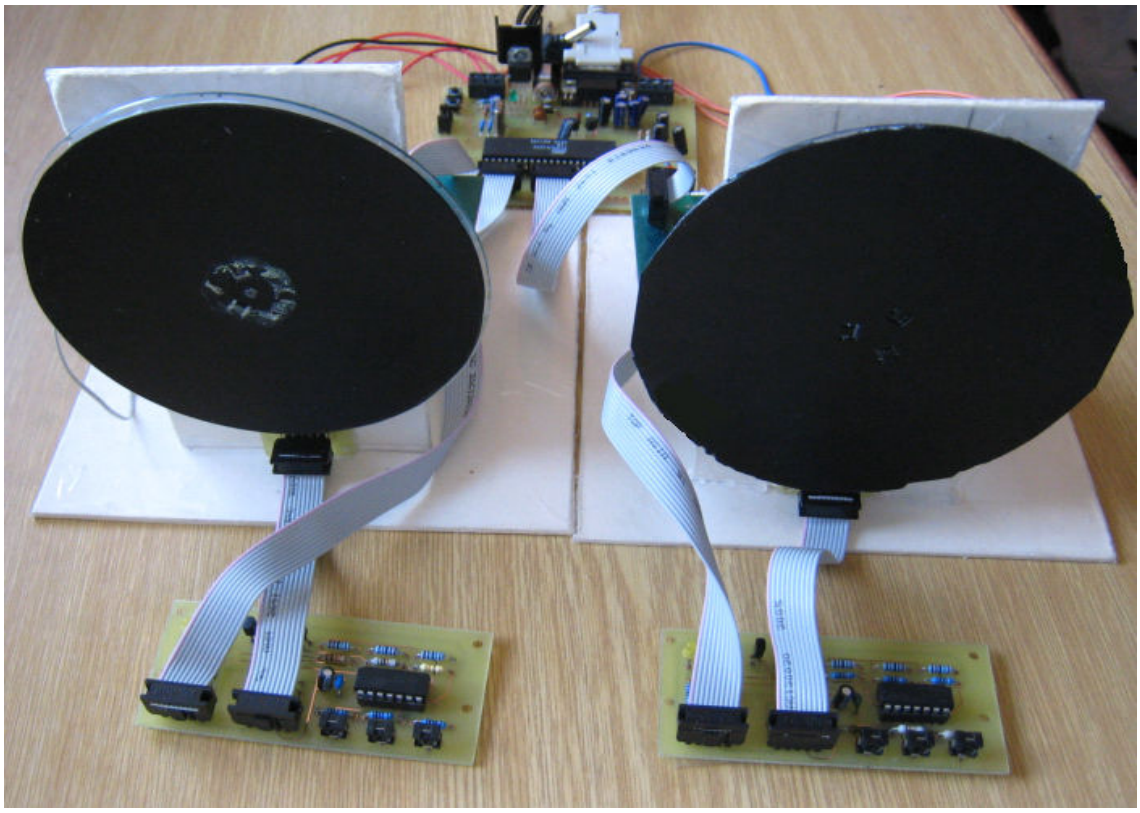
A, Fotky hardwaru



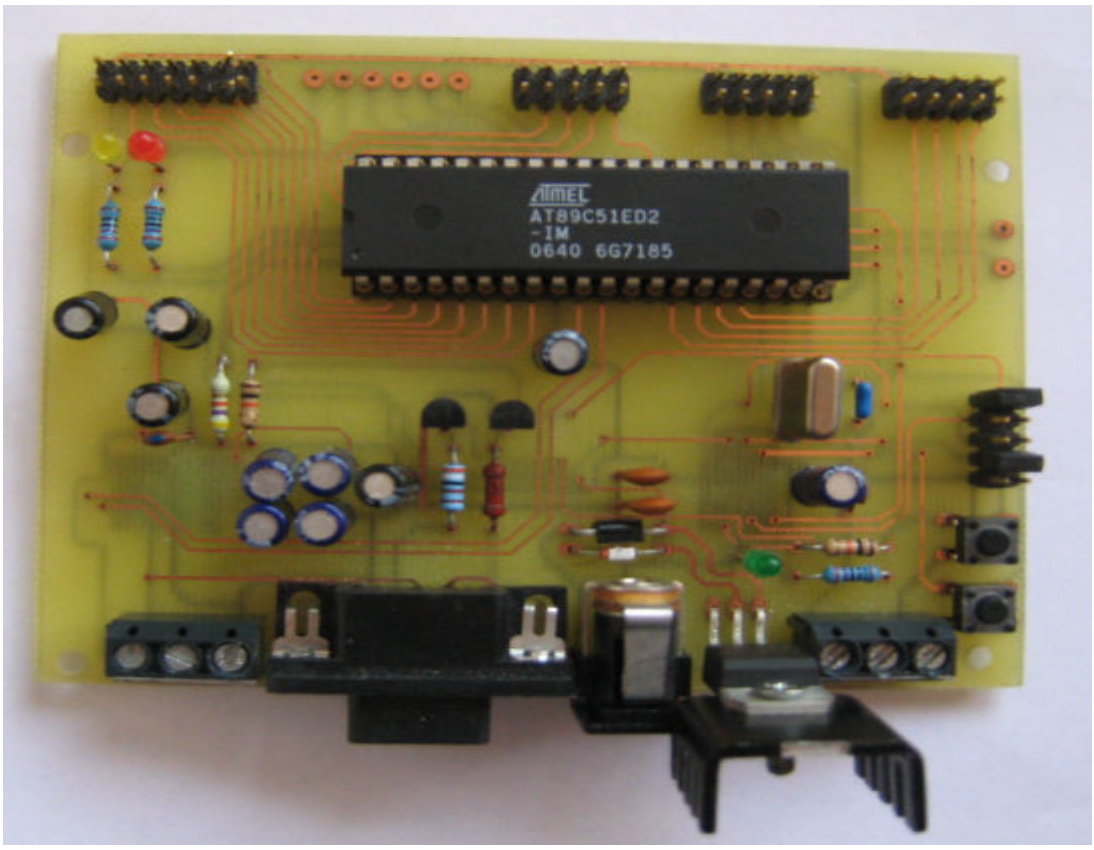
Obr.A1 Náhled na celkové zapojení



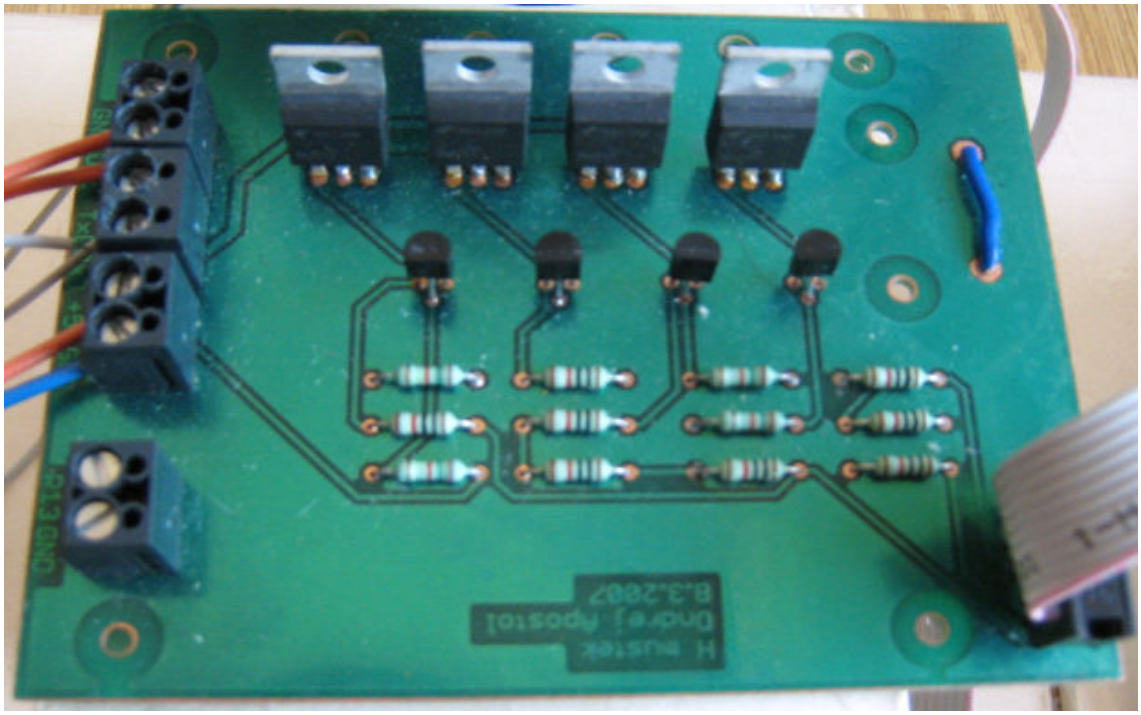
Obr.A2 Náhled na PŘJ a H můstky



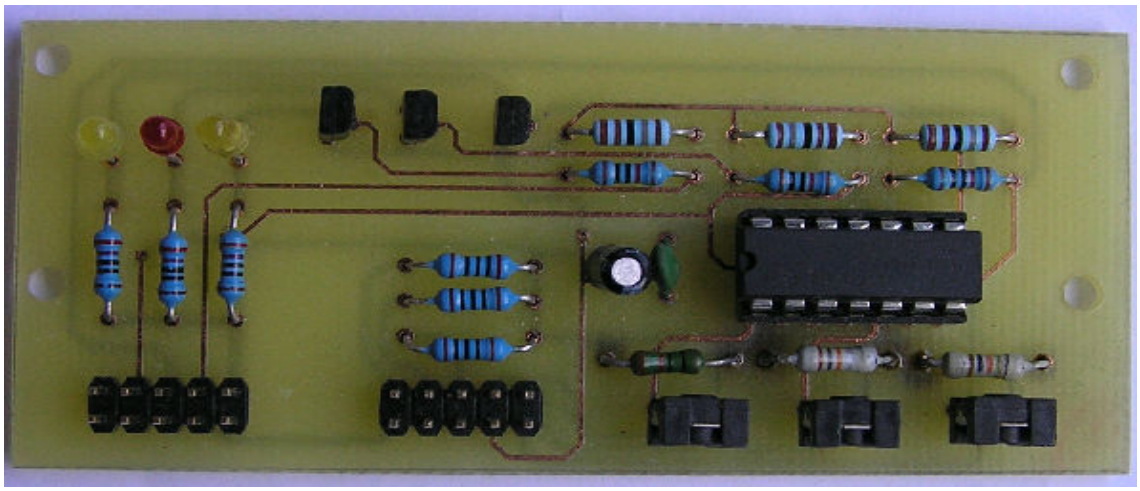
Obr.A3 Čelní pohled



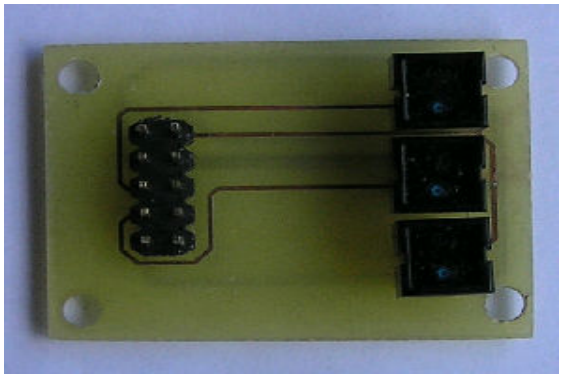
Obr.A4 Programovatelná řídící jednotka



Obr.A5 H můstek

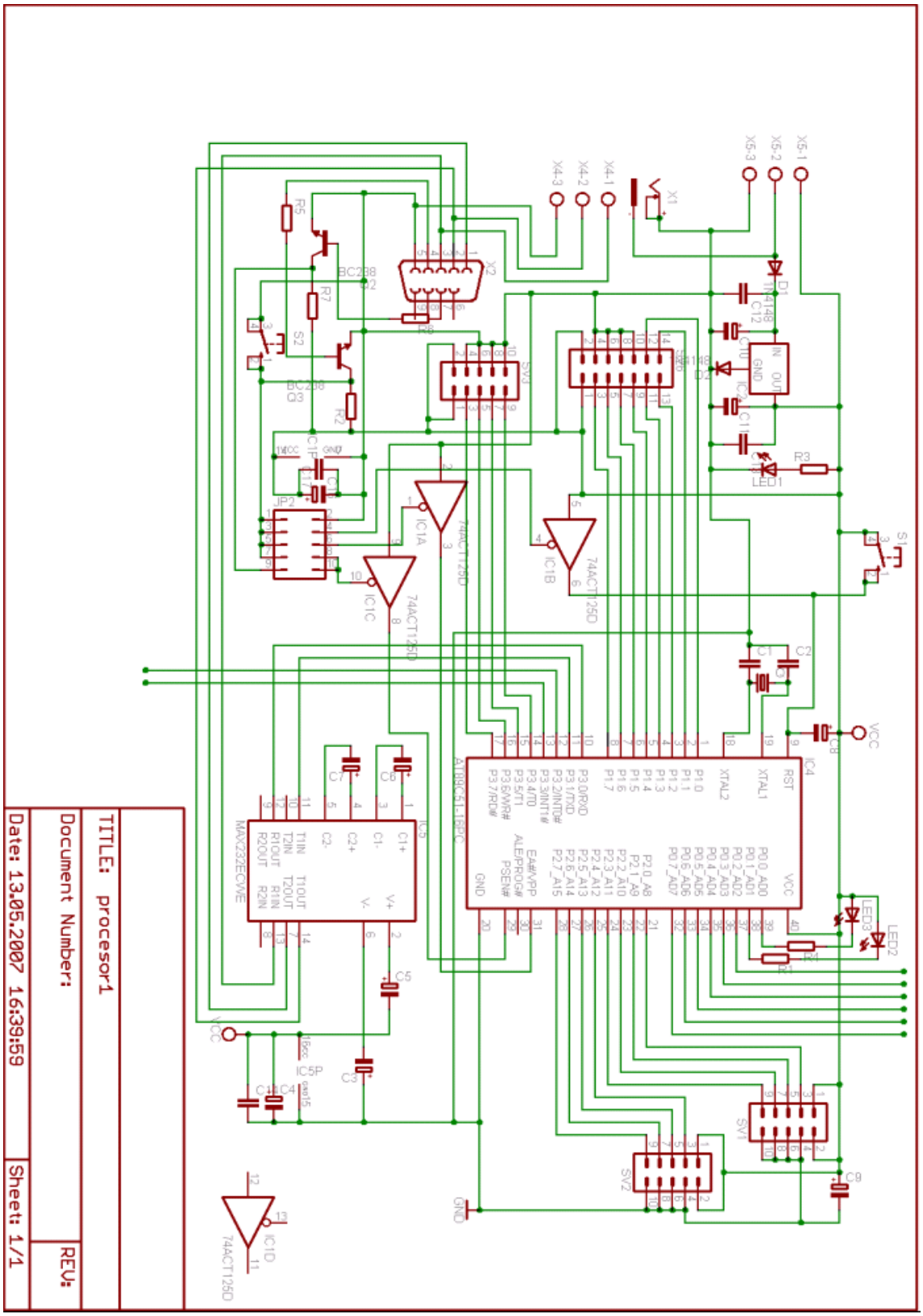


Obr.A6 Ovládání čidel

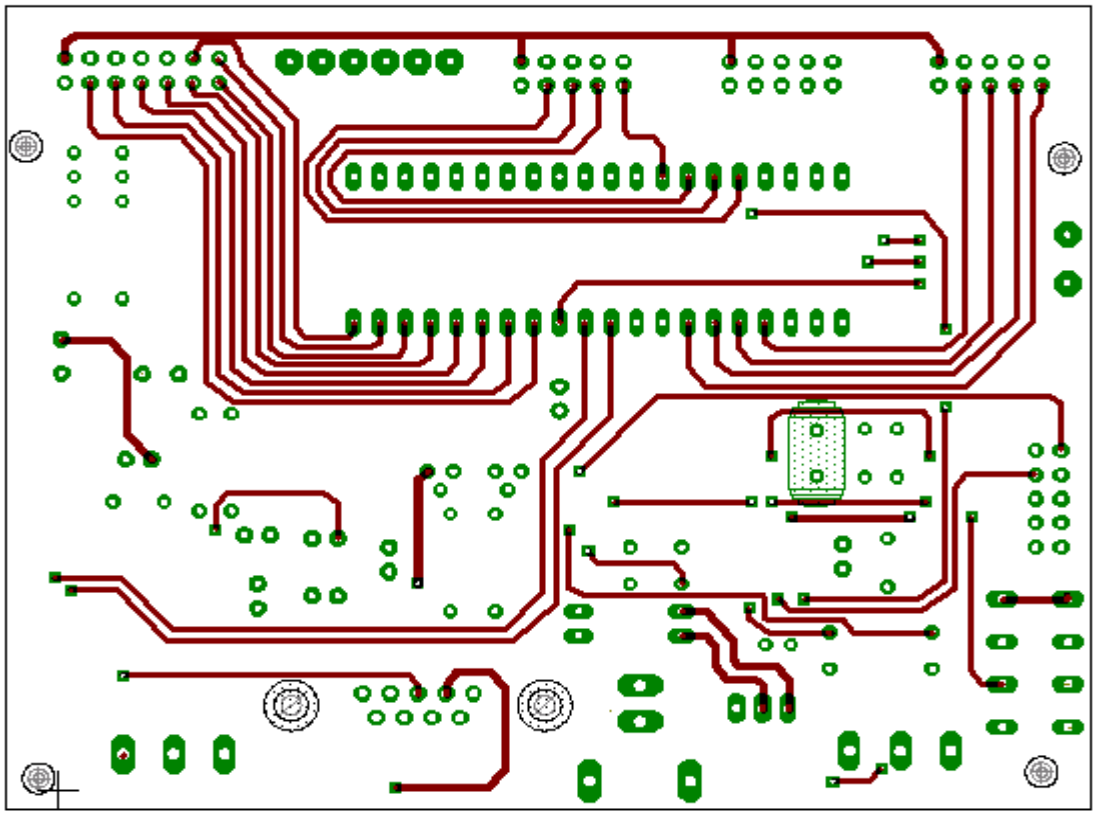


Obr.A7 Čidla

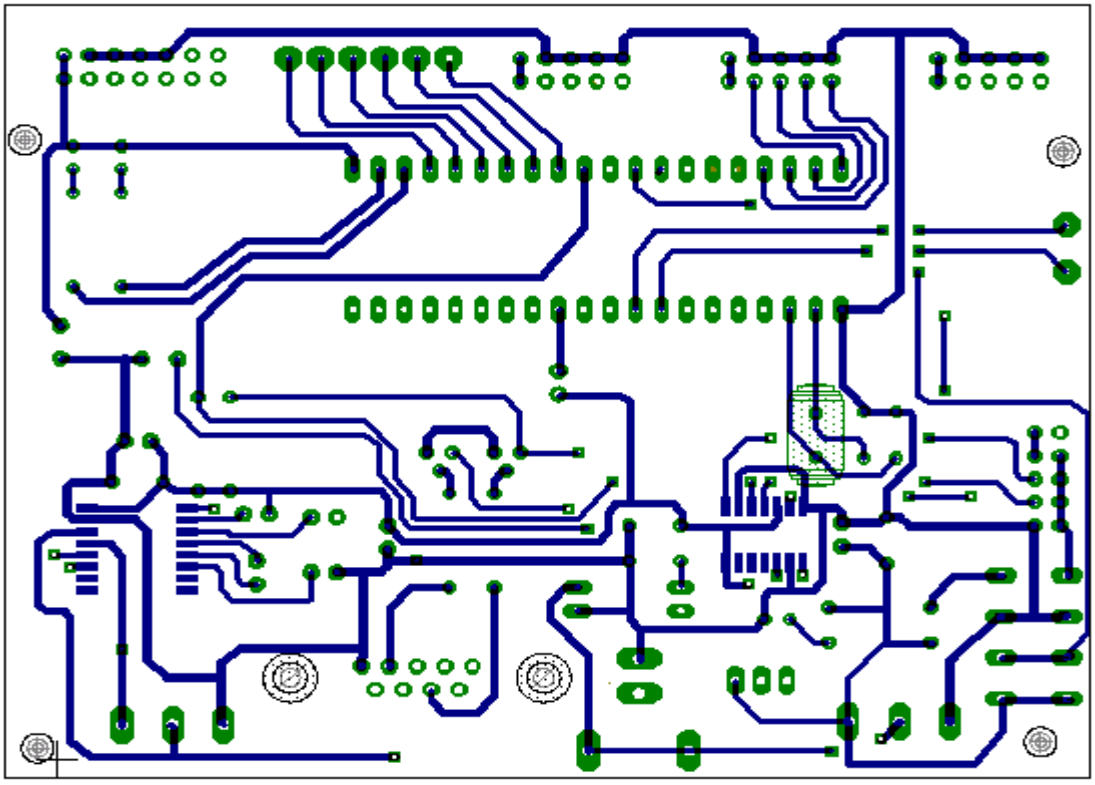
B, Dokumentace



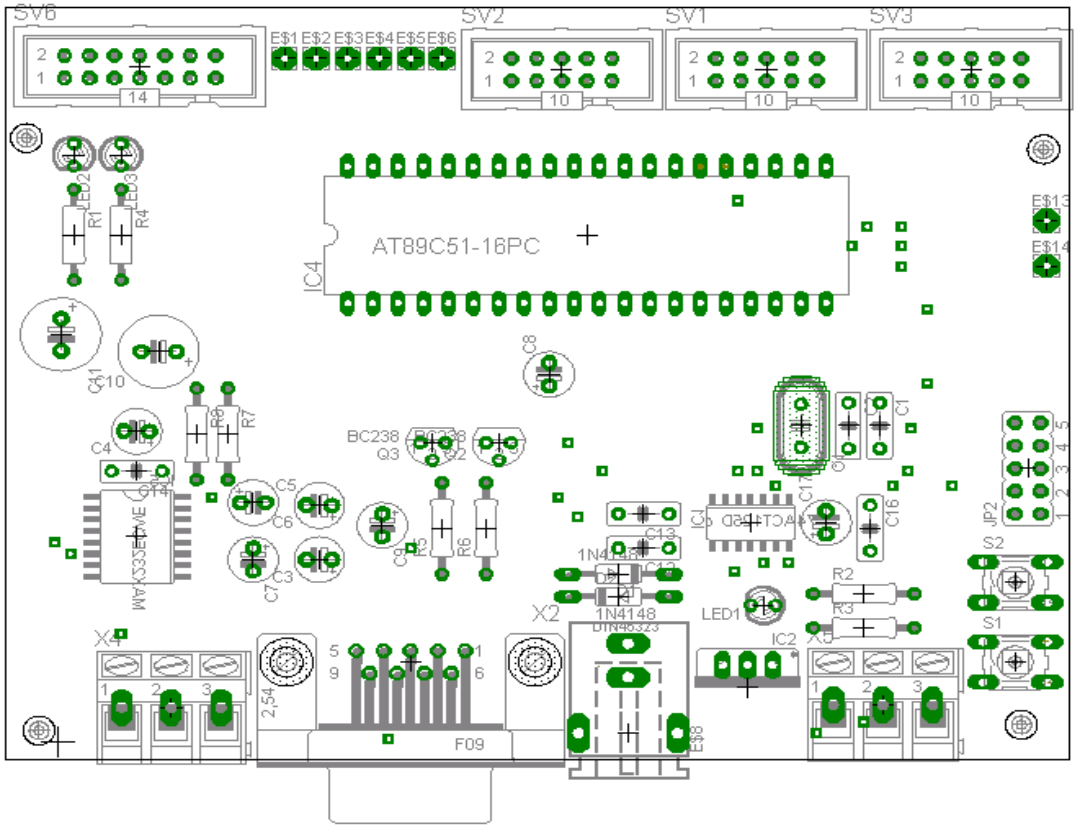
Obr.B1-1 Schéma PŘJ



Obr.B1-2 Deska plošného spoje PŘJ vrchní část



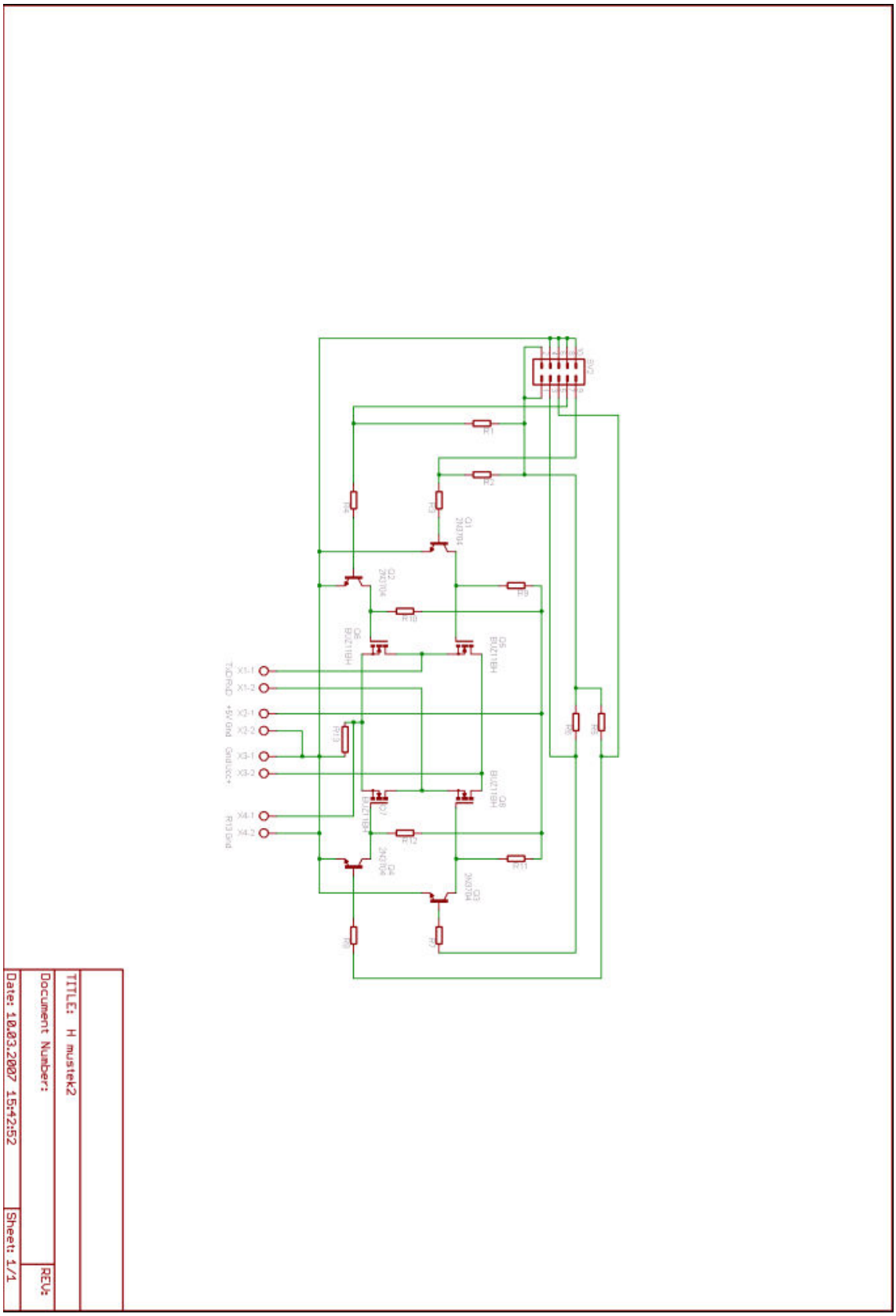
Obr.B1-3 Deska plošného spoje PŘJ spodní část



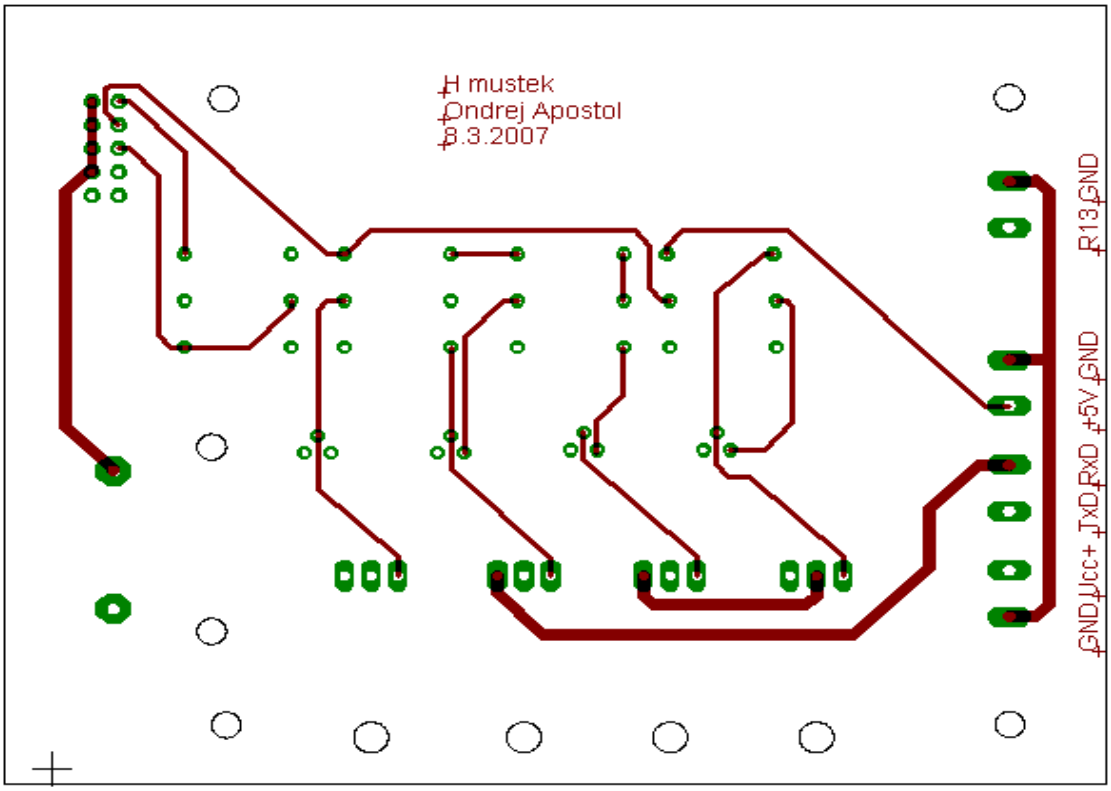
Obr.B1-4 Osazovací plán PŘJ

Tab.B1 Použité součástky PŘJ (výpis součástek z programu EAGLE)

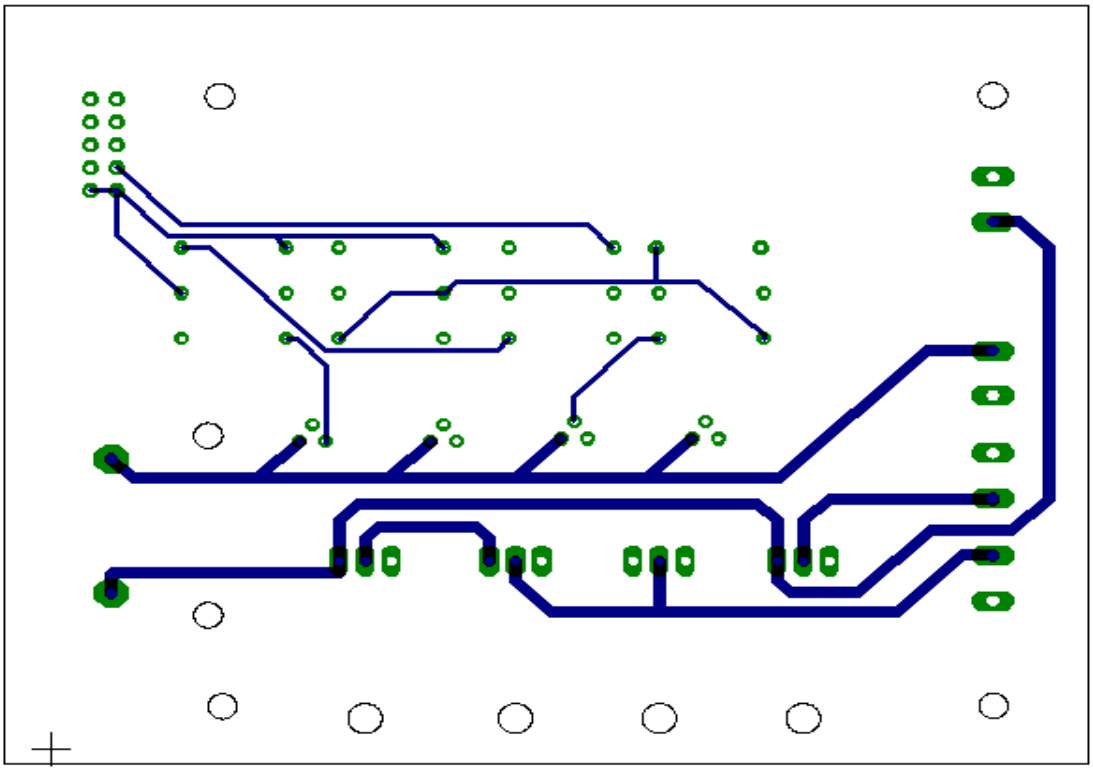
Označení součástky	Pouzdro	Počet
C1	C-EU050-025X075	1
C2	C-EU050-025X075	1
C3	CPOL-EUE2.5-5	1
C4	CPOL-EUE2.5-5	1
C5	CPOL-EUE2.5-5	1
C6	CPOL-EUE2.5-5	1
C7	CPOL-EUE2.5-5	1
C8	CPOL-EUE2.5-5	1
C9	CPOL-EUE2.5-5	1
C10	CPOL-EUE2.5-5	1
C11	CPOL-EUE2.5-5	1
C12	C-EU050-025X075	1
C13	C-EU050-025X075	1
C14	C-EU050-025X075	1
C15	C-EU050-025X075	1
C16	C-EU050-025X075	1
C17	CPOL-EUE2.5-5	1
D1	1N4148	1
D2	1N4148	1
IC1	74ACT125D	1
IC2	78XXS	1
IC4	AT89C51-16PC	1
IC5	MAX232ECWE	1
JP2	JP5Q	1
LED1	LED3MM	1
LED2	LED3MM	1
LED3	LED3MM	1
Q1	CRYTALHC49U-V	1
Q2	BC238	1
Q3	BC239	1
R1	R-EU_0207/10	1
R2	R-EU_0207/10	1
R3	R-EU_0207/10	1
R4	R-EU_0207/10	1
R5	R-EU_0207/10	1
R6	R-EU_0207/10	1
R7	R-EU_0207/10	1
S1	10-XX	1
S2	10-XX	1
SV1	ML10	1
SV2	ML10	1
SV3	ML10	1
SV6	ML10	1
X1	737992-55	1
X2	F09H	1
X4	AK300/3	1
X5	AK300/3	1



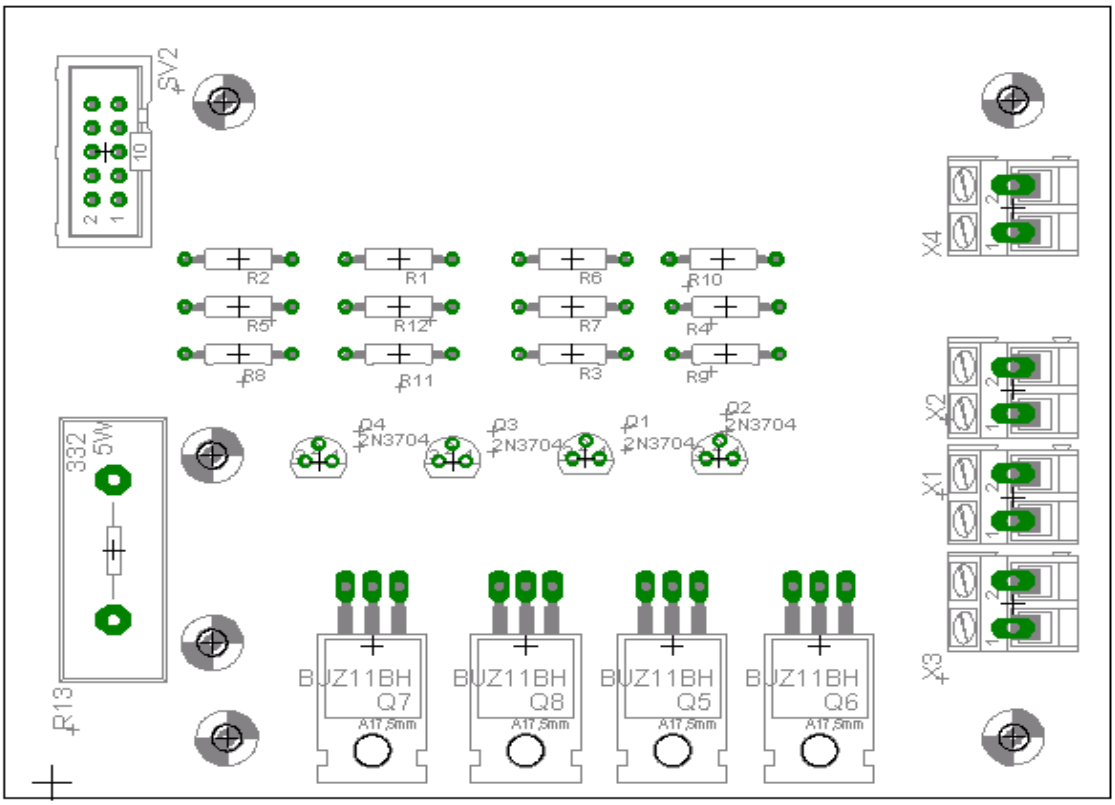
Obr.B2-1 Schéma h můstku



Obr.B2-2 Deska plošného spoje h můstku vrchní část



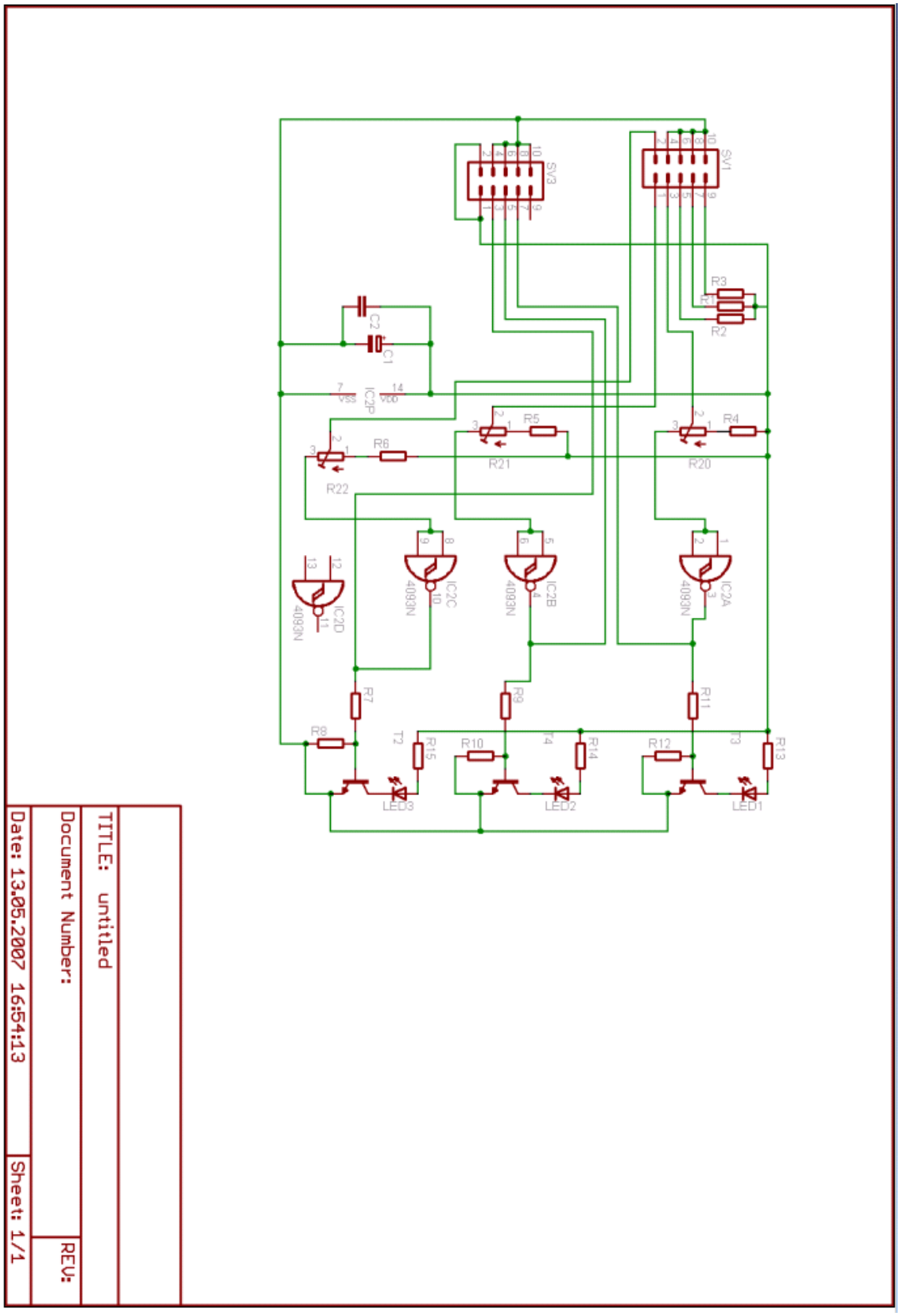
Obr.B2-3 Deska plošného spoje h můstku spodní část



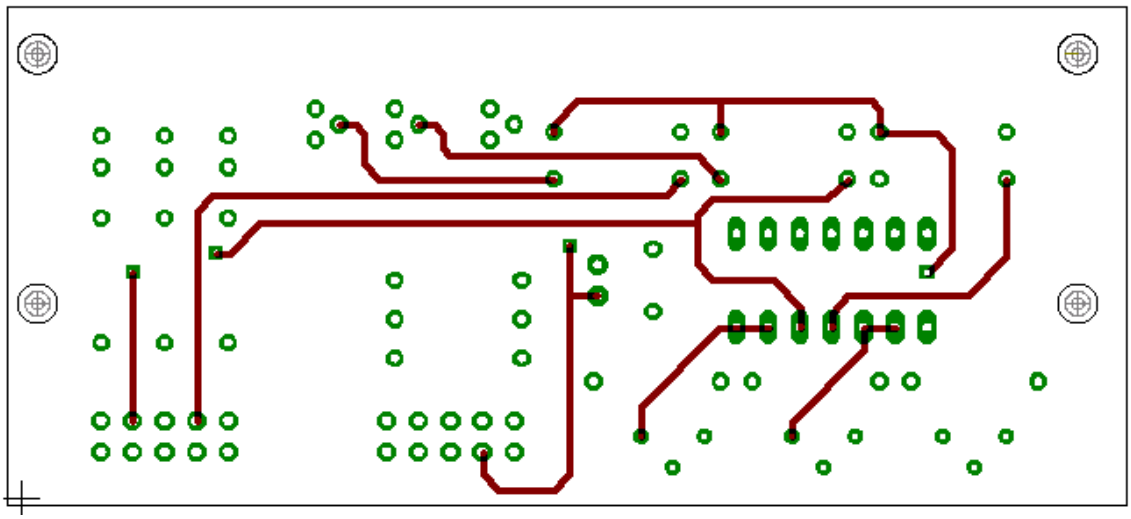
Obr.B2-4 Osazovací plán h můstku

Tab.B2 Použité součástky h můstku

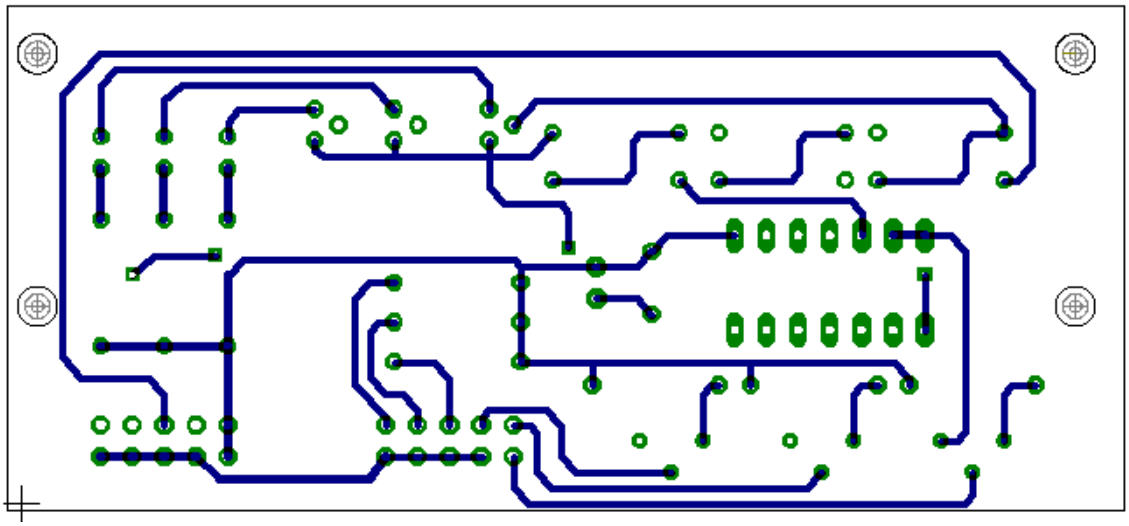
Označení součástky	Součástka	Počet
Q1	2N3704	1
Q2	2N3705	1
Q3	2N3706	1
Q4	2N3707	1
Q5	BUZ11BH	1
Q6	BUZ11BH	1
Q7	BUZ11BH	1
Q8	BUZ11BH	1
R1	R-EU_0207/10	1
R2	R-EU_0207/10	1
R3	R-EU_0207/10	1
R4	R-EU_0207/10	1
R5	R-EU_0207/10	1
R6	R-EU_0207/10	1
R7	R-EU_0207/10	1
R8	R-EU_0207/10	1
R9	R-EU_0207/10	1
R10	R-EU_0207/10	1
R11	R-EU_0207/10	1
R12	R-EU_0207/10	1
R13	R-EU_0207/10	1
SV2	R-EU_0207/10	1
X1	AK300/2	1
X2	AK300/2	1
X3	AK300/2	1
X4	AK300/2	1



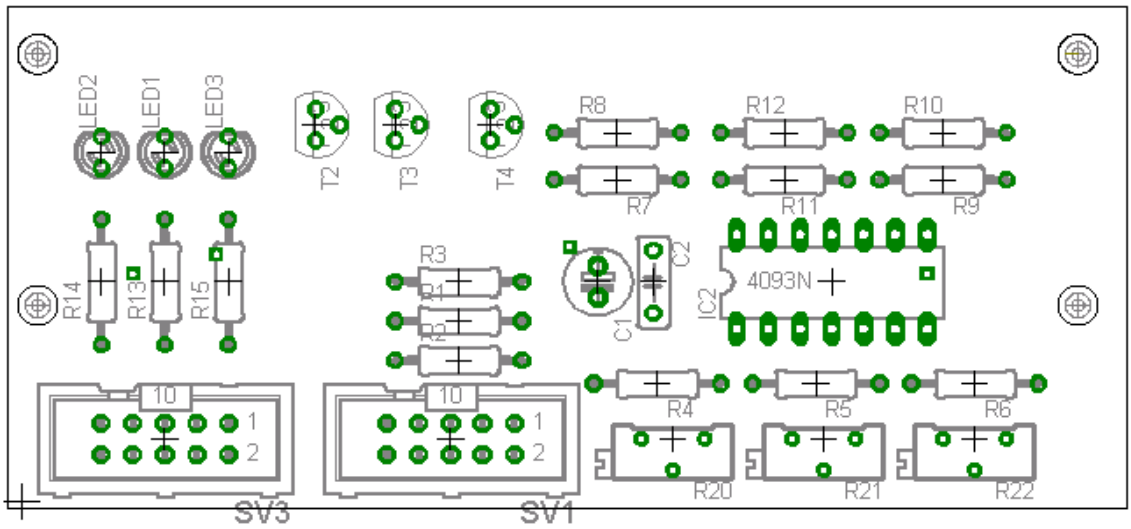
Obr.B3-1 Schéma ovládání čidel



Obr.B3-2 Deska plošného spoje ovládání čidel vrchní část



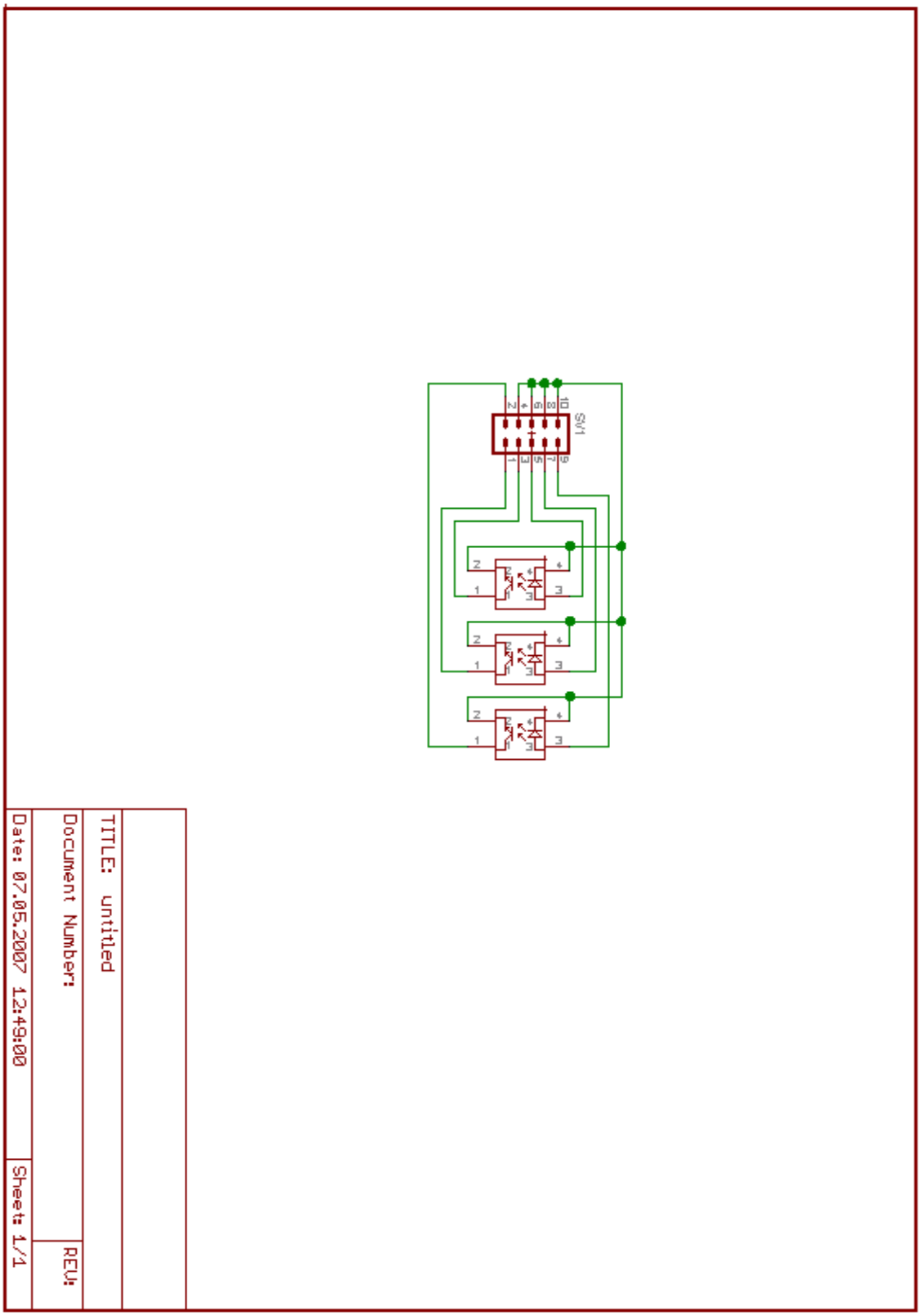
Obr.B3-3 Deska plošného spoje ovládání čidel spodní část



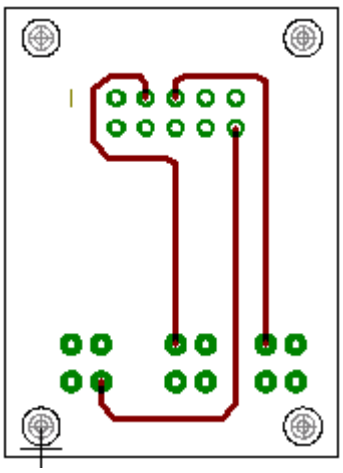
Obr.B3-4 Osazovací plán ovládání čidel

Tab.B3 Použité součástky ovládání čidel

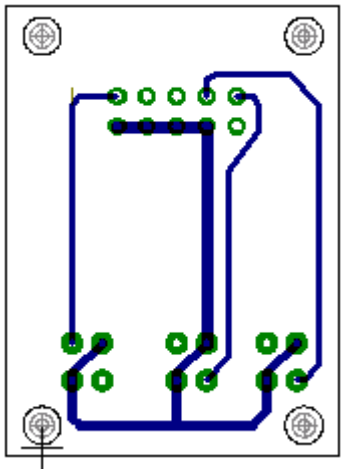
Označení součástky	Součástka	Počet
C1	E2,5-6E	1
C2	C050-025X075	1
IC2	DIL14	1
LED1	LED3MM	1
LED2	LED3MM	1
LED3	LED3MM	1
R1	0207/10	1
R2	0207/10	1
R3	0207/10	1
R4	0207/10	1
R5	0207/10	1
R6	0207/10	1
R7	0207/10	1
R8	0207/10	1
R9	0207/10	1
R10	0207/10	1
R11	0207/10	1
R12	0207/10	1
R13	0207/10	1
R14	0207/10	1
R15	0207/10	1
R20	RTRIM64Z	1
R21	RTRIM64Z	1
R22	RTRIM64Z	1
SV1	ML10	1
SV3	ML10	1
T2	TO92	1
T3	TO92	1
T4	TO92	1



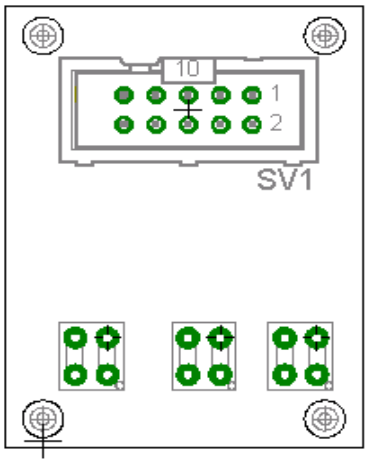
Obr.B4-1 Schéma zapojení čidel



Obr.B4-2 DPS zapojení čidel vrchní část



Obr.B4-3 DPS zapojení čidel spodní část



Obr.B4-4 Osazovací plán zapojení čidel

Tab.B4 Použité součástky zapojení čidel

Označení součástky	Součástka	Počet
SV1	ML10	1
U$1	INFRA	1
U$2	INFRA	1
U$3	INFRA	1